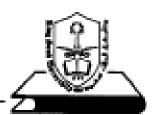
استخدام برنامج MATLAB في الرياضيات الجامعية

تأليف

د. عبير حميدي الحربي

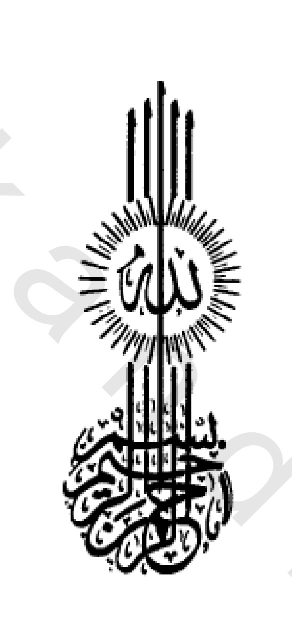
الأستاذ المشارك بقسم الرياضيات – كلية العلوم

جامعة الملك سعود



النشر العلمي والمطابع - جامعة الملك سعود

ص.ب ٦٨٩٥٣ - الرياض ١١٥٣٧ - المملكة العربية السعودية


بسم الله الرحمن الرحيم

فهرسة مكتبة الملك فهد الوطنية أثناء النشر

الحربي، عبير حميدي

استخدام برنامج MATLAB في الرياضيات الجامعية . / عبير حميدي الحربي .

الرياض، ١٤٣٢هـ.

٢٧٨ ص، ١٧ × ٢٤سم

ردمك : ٥ – ٧٩١ – ٥٥ – ٩٩٦٠–٩٧٨

١- الرياضيات – معالجة البيانات أ. العنوان

ديوي ٢٨٥ , ٥١٠ ١٤٣٢/٣٥٧٥

رقم الإيداع ١٤٣٢/٣٥٧٥

ردمك : ٥ – ٧٩١ – ٥٥ – ٩٩٦٠–٩٧٨

حكمت هذا الكتاب لجنة متخصصة، شكلها المجلس العلمي بالجامعة، وقد وافق المجلس على نشره – بعد إطلاعه على تقارير المحكمين– في اجتماعه الثاني للعام الدراسي ١٤٣٢/١٤٣١هـ المعقود في تاريخ ١٤٣١/١٠/٢٤هـ الموافق ٢٠١٠/١٠/٣م.

النشر العلمي والمطابع ١٤٣٢هـ

مقدمـــة

الحمد لله رب العالمين والصلاة والسلام على رسوله الكريم محمد صلى الله عليه وعلى آله وصحبه وسلم وبعد :

إن علم الرياضيات أسهم بشكل واسع في تطور العلوم كافة، وفي بعض مجالات الحياة المختلفة. والرياضيات ليست فقط أداة تنمي الفكر وتطوره، بل هي علم مهم وأساسي في التخصصات العلمية والإدارية، لذلك ظهرت برمجيات تستخدم الحاسب الآلي لتوفّر الوقت وسرعة الإنجاز، والدقة في إجراء العمليات الحسابية، والقدرة على التعامل مع كمٍّ من البيانات. وفي مقدمة هذه البرمجيات برنامج MATLAB، الذي يمكّن المختص في العلوم من تطوير مهاراته من خلال عدد هائل من الأمثلة و المسائل بسرعة متناهية ودقة عالية. كما أنه يوفر حلولاً رياضية للعاملين في مشاريع علمية وهندسية، بقدرات فائقة وإمكانات للنمذجة والمحاكاة.

لقد عزمتُ على تأليف هذا الكتاب نظراً لطلب العديد من الزملاء والطلاب من التخصصات العلمية المختلفة لمرجع باللغة العربية يشرح أساسيات استخدام MATLAB في المواضيع الرياضية المختلفة، لعله يشكل إضافة علمية للمكتبة العربية، وخطوة متقدمة نحو تطوير صيغ التعليم التقليدي إلى صيغ التعليم الحديث التي تسخّر إمكانات الحاسب الآلي. وهذا الكتاب ليس بديلاً لتعلم مفاهيم الرياضيات، إلا أنه مكمل ومستخدم لها، فالمفاهيم تُعرض بشكل مبسط، ويُركزعلى عرض تطبيقاتها

باستخدام الحاسوب، دون اللجوء للحل الرياضي المعتاد. ينقسم الكتاب إلى سبعة فصول، في نهاية كلٍ منها بعض التمارين التي تساعد القارئ على تطبيق الأوامر الجديدة. كما يحتوي الكتاب على العديد من الأمثلة والخوارزميات، ويفترض في القارئ أن يكون على إلمامٍ بمبادئ الجبر الخطي والتحليل العددي وأساسيات البرمجة.

فالفصل الأول يحتوي على مقدمة تعريفية لبرنامج MATLAB وأهم أوامره و خصائصه وأساسيات برمجة الخوارزميات البسيطة. أما الفصلان الثاني والثالث فيتضمّنان مفاهيم من الجبر الخطي كحلول المعادلات الرياضية والأنظمة الخطية وغير الخطية، وبالنسبة للفصل الرابع فهو يغطي بعض مواضيع حساب التفاضل والتكامل، بينما يقدم الفصل الخامس حلولاً عددية للمعادلات التفاضلية. ويتطرّق الفصل السادس للاستكمال والتقريب، أما الفصل السابع والأخير فقد خصصتُه لتقديم مواضيع رياضية متفرقة، مثل جبر المتجهات، والطرق المثلى، ودوال الإحصاء وعلم التعمية.

هذا ولا يفوتني أن أتوجه بالشكر إلى طالبات مقرر مشروع البحث في قسم الرياضيات بجامعة الملك سعود، وكذلك زملائي في الجامعة الذين كان لهم دور كبير في اختيار المواضيع التي طُرحت في الكتاب وفي التعديلات التي طرأت على الأمثلة. وفي الختام آمل أن أكون قد وفّقت في تقديم نبذة عن برنامج MATLAB وبعض تطبيقاته الرياضية المختلفة، وأن يجد هذا الكتاب الاستحسان و القبول لدى القارئ. كما أرحب بالآراء والنقد البناء من الزملاء والطلاب وذلك على البريد الإلكتروني abir@ksu.edu.sa .

سائلةً الله أن يجعل عملي خالصاً لوجهه تعالى، وأن يوفق الجميع لخدمة التقدم العلمي من خلال إثراء المكتبة العربية ، والله ولي التوفيق.

المؤلفـــة

المحتويات

Contents

الصفحة

الفصل الرابع: حساب التفاضل والتكامل في MATLAB

الفصل الخامس: حلول المعادلات التفاضلية على MATLAB

الفصل السادس: استكمال وتقريب الدوال على MATLAB

الفصل السابع: مواضيع رياضية متفرقة على MATLAB

الملحق

ثبت المصطلحات

الفصل الأول

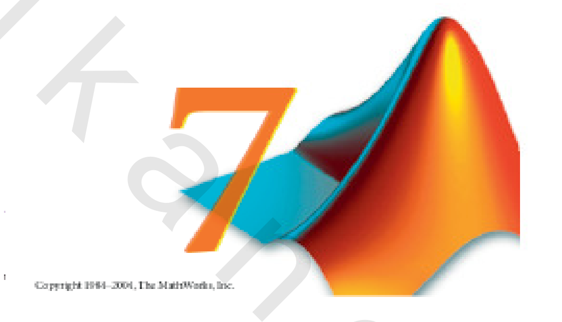
مبادئ برنامج MATLAB

(١,١) مقدمة في MATLAB

MATLAB هو برنامج عالي الأداء، صُمّم لإجراء الحسابات الرياضية المتقدمة، ويتميز بكونه برنامجاً متخصصاً ييسّر عمل الباحثين والدارسين في مختلف مجالات الدراسات العليا والدراسة الجامعية، ويضمّ المئات من الدوال الجاهزة التي توفر للمبرمج وقتاً وجهداً عند كتابة البرامج. أصول MATLAB ترجع إلى محاولة الرياضيين لتسهيل حساب المصفوفات، ومن هذا المنطلق فإن معنى كلمة MATLAB هو معمل المصفوفات "Matrix Laboratory" و قد بدأ تطويره من قبل شركة MathWorks في عام ١٩٨٤م ، مع أن تصميمه الأول كان على يد العالم Cleve Moler في ١٩٧٠م، الذي كان يشغل منصب رئيس قسم علوم الحاسب في جامعة نيو ميكسيكو بالولايات المتحدة الأمريكية. يوجد لبرنامج MATLAB عدة نسخ صدرت على مدى أعوام، مثل النسخة 5 MATLAB و 6 و 7. نستخدم هنا في عرضنا النسخة السابعة 7 MATLAB الشكل رقم (١,١). وهو يعمل على كل أنظمة Windows 95/98، Windows 2000/NT، Windows XP ، كما يوجد إصدار مماثل يخصّ Macintosh و Unix. ويمكن لـ MATLAB التعامل مع مستخدم واحد single user ،

أو أكثر من مستخدم للشبكات Workstation ، وهو عبارة عن حزمة من البرمجيات الجاهزة (Toolboxes) يحصل عليها المستخدم بحسب الحاجة.





الشكل رقم (١،١). الإصدار السابع من MATLAB.

إن برنامج MATLAB عمليٌّ جداً، وخاصة في تمثيل البيانات بالرسم سواء منحنيات أو مسطحات، ويمكّن المستخدم من التحكم بالرسم بسهولة. و رغم أن MATLAB لغة متطورة في البرمجة إلا أنه من الممكن للمبرمج المبتدئ التعامل معه ببساطة. ويحتوي MATLAB على توابع و دوال جاهزة لتغطية أكثر المسائل شيوعاً. فمثلاً تتطلب كل من لغة Fortran ، C و Pascal عشرات الأسطر لكتابة برنامج لإيجاد تحويل فورير Fourier Transform بينما لدى MATLAB أمر ضمني واحد (fft) لإنجاز هذه العملية. وكذلك فإن عدم تعريف المتغيرات وحجمها مسبقاً من

المميزات لـ MATLAB على باقي لغات البرمجة، ولكنه أبطأ لأنه يقوم بترجمة البرنامج المصدري جملة جملة interpative language بينما تقوم اللغات Fortran، Pascal و C بترجمة البرنامج المصدري كاملاً وتحويله إلى لغة الآلة compiled language ، لذلك يتم تنفيذه بطريقة أسرع. فبرنامج MATLAB يمكن الاستفادة منه بطريقتين، إما للحسابات المكثفة بالدوال الجاهزة، أو بالبرمجة مع إدراج الدوال الجاهزة لتبسيط و تسريع أداء البرامج.

برنامج MATLAB يتعلق بالحسابات الرياضية، والهندسية والمحاكاة. ويستخدم في الصناعات المختلفة، كما يستخدم للأغراض الأكاديمية، وخصوصاً أغراض البحث العلمي في الغالبية العظمى من جامعات العالم. والكثير من شركات الطيران المدني والعسكري، تستخدم MATLAB في الحسابات الهندسية، والنمذجة والمحاكاة، مثل شركة إيرباص. كما يُعتمد عليه في تصميم الطائرات التي تطير بدون طيّار، و في أبحاث الفضاء لشركة ناسا في مجال معالجة البيانات من قبل الباحثين والمختصين ليتمكن الباحث من إجراء مئات التجارب، الأمر الذي يستحيل فعله بطريقة يدوية. وهو الآن شائع الاستخدام في التدريس خاصة في مواد الجبر الخطي، والتحليل العددي وتطبيقاته المختلفة.

يهدف الكتاب إلى عرض بعض التطبيقات الرياضية على MATLAB لمستوى المقررات الجامعية الأولى، و ذلك لمساعدة طالب العلوم الطبيعية أو الهندسة على ترسيخ المفاهيم الرياضية، خاصةً التي تعتمد على التحليل العددي والحسابات العددية المكثفة. يحتاج القارئ إلى الإلمام بمبادئ البرمجة البسيطة والطرق الرياضية المختلفة، وذلك لأن الكتاب يركز في عرضه على كيفية الاستخدام الجيد لبرنامج MATLAB من أجل كتابة برامج توضح هذه المفاهيم المختلفة. كما أن الكتاب يوضح طرقاً مختلفة

لتسخير MATLAB كآلة مساعدة لتوضيح المفهوم الرياضي، بنفس الطريقة التي تُستخدم فيها الآلة الحاسبة أو الأدوات الهندسية في إقناع الطالب بصحة القواعد والنتائج. وبعد الاطلاع على الكتاب وأداء التطبيقات، أدعو القارئ للاكتشاف بنفسه قوة MATLAB عن طريق التطبيقات المتخصصة في مجاله. من الصعب ذكر كل دوال MATLAB واستخداماتها الرياضية في كتاب واحد، وإنما نحتاج إلى عدة كتب لتغطية ذلك، و لو بحث القارئ في أيّ محرك بحث على الإنترنت عن مفهوم رياضي وتطبيقاته باستخدام MATLAB لحصل على عشرات المواقع التي تتحدث عن أمثلة جديدة وتطبيقات مبتكرة. يحتوي الكتاب الحالي على التطبيقات الرياضية الأساسية والتي تعطي الخطوط العريضة guidelines لقدرات MATLAB وتساعد القارئ على استخدامها وتسخيرها لتطبيقاته المختلفة.

(١,٢) الأوامر الرئيسة في MATLAB

يُعدّ MATLAB واحداً من أهم البرامج التي تقدم حلولاً متكاملة في مجال الرياضيات و الاختصاصات المعتمدة عليها والتي لا حصر لها، وهو يوفر للمستخدم:

١- التعامل بسهولة مع المصفوفات وحساباتها.

٢- عدداً كبيراً من الدوال الجاهزة والبرامج المخزنة، وهي في تطوير دائم مع كل نسخة جديدة.

٣- الرسم المتقن في بعدين و ثلاثة أبعاد.

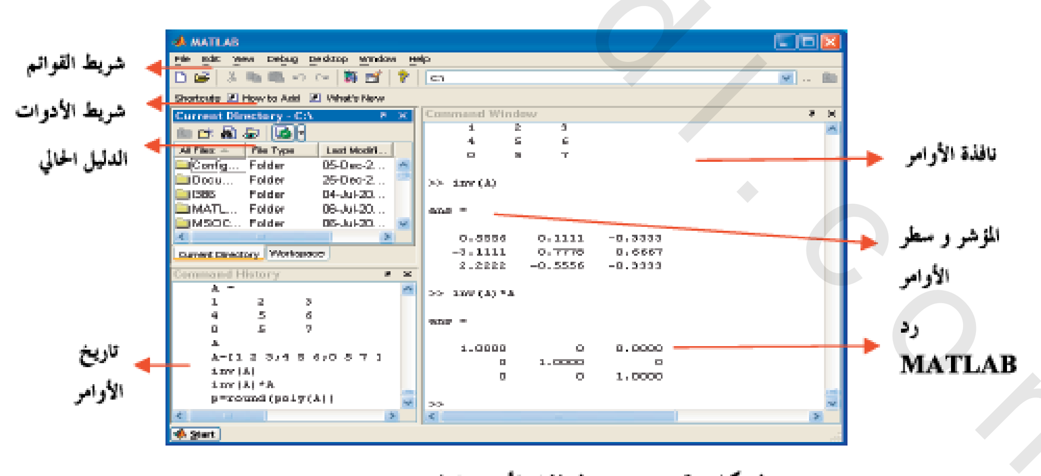
٤- القدرة على البرمجة للاستفادة من MATLAB في كافة المجالات العلمية.

عند فتح برنامج MATLAB تظهر عدة نوافذ مختلفة تظهر في الشكل رقم (١,٢) وهي:

١- **نافذة الأوامر الحالية Command window**: وهي النافذة الأساسية التي تتخاطب من خلالها مع برنامج MATLAB وهي تشمل Workspace مجال العمل أو الذاكرة المؤقتة، حيث يتم فيها حفظ جميع المتغيرات التي استُعملت إلى حين إغلاق MATLAB ما لم يتم تنفيذ الأمر clear.

٢- **نافذة تاريخ الأوامر Command History**: و تعرض جميع الأوامر التي كتبت في مجال العمل Workspace الحالي.

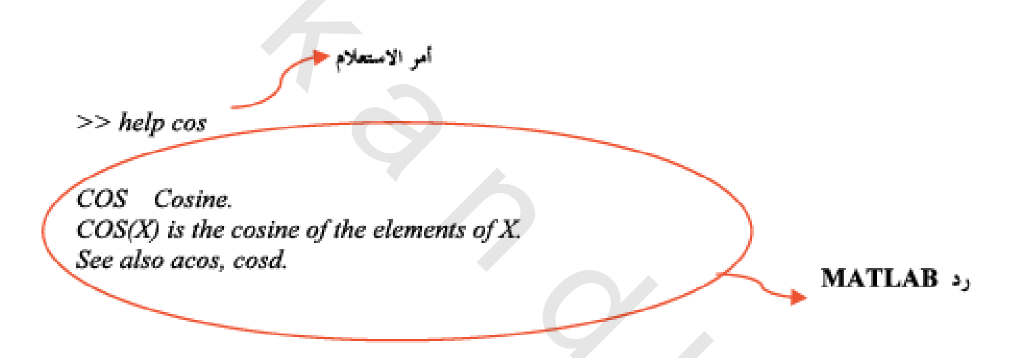
٣- **الدليل الحالي Current Directory**: و يعرض موقع الملف الحالي لـ Workspace.



الشكل رقم (١,٢). النافذة الأساسية في MATLAB.

أهم الأوامر الخاصة بالنافذة الأساسية

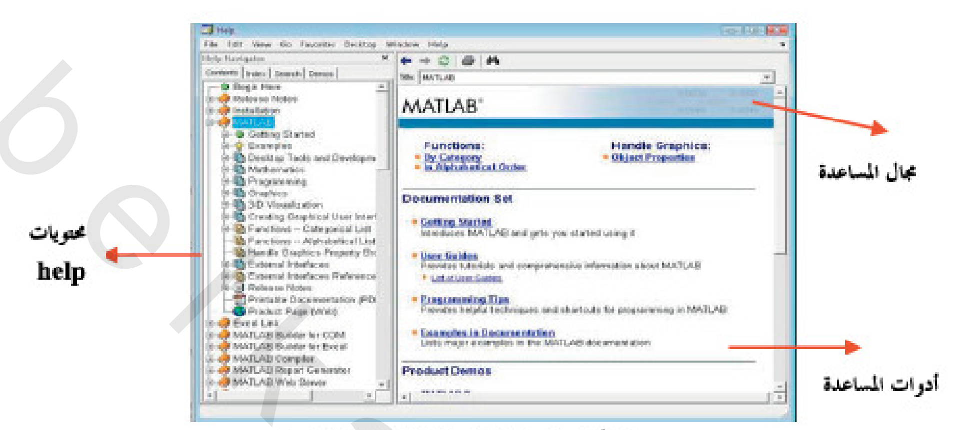
• يُعدّ الأمر *help* المستخدم للمساعدة الطريقة الأساسية لمعرفة صيغ الدوال وتطبيقاتها، ويقوم بعرض المعلومات الضرورية مباشرة في نافذة الأوامر، ولاستخدامه اكتب الأمر *help* عند المؤشر >> ، وبعدها بمسافة اكتب اسم الدالة المطلوبة للاستعلام، ثم اضغط زر الإدخال Enter . وسوف يقوم MATLAB بالرد على هذا الاستعلام بشرح بسيطٍ للدالة كما هو موضح في التالي.



كذلك توجد نافذة help المبينة في الشكل رقم (١.٣) والتي يتم فتحها من شريط المهام في MATLAB .

• الأمر *who* يعرض جميع متغيرات workspace في command window.

• الأمر *clear all* يقوم بمسح جميع محتويات workspace ويمكن مسح المتغير x فقط بالأمر *clear x* أو مسح المتغيرات التي تبدأ بحرف a فقط *clear a**.



الشكل رقم (١,٣). نافذة المساعدة **help**.

- للقيام بحفظ كل محتويات workspace في ملف اسمه mywork ندخل الأمر *save mywork* أما إذا أردنا حفظ متغير مثل x نكتب *save x*.
- ولاسترجاع الملف mywork ندخل الأمر *load mywork* ، ولاسترجاع متغير x نكتب *load x* .
- العلامة ; بعد أي أمر تمنع طباعة النتائج المحسوبة.
- العلامة % تسبق كل جملة تعليق comment line أي أن MATLAB يتجاهل كل ما كتب من أوامر في هذه الجملة.
- العلامة ... في نهاية السطر تعني تابع الجملة في السطر الجديد.

مثال رقم (١,١)

ندخل بعض من هذه الأوامر :

```
>> % some examples
>> x=4
x =
    4
>> who
Your variables are:
x
>> save x
>> load x
>> clear x
```

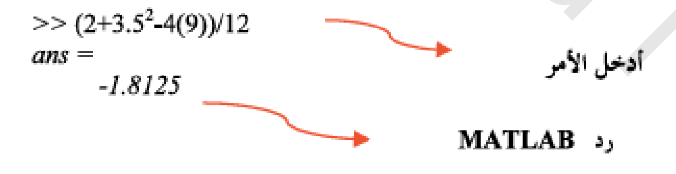
تم تخزين المتغير x

تم التحميل

مسح المتغير x

(١,٣) الحسابات البسيطة في MATLAB

عند تشغيل برنامج MATLAB تظهر العلامة >> وتسمى Command Line ويمكن إدخال أمر بعدها ، وبعد ضغط ENTER يتم تنفيذ الأمر ، و يظهر رد MATLAB بعد عبارة ans . فمثلاً لحساب العبارة الرياضية البسيطة $(2+3.5^2-4(9))/12$.



ويمكن تعريف متغيرات لحساب العبارة:

```
>> x=2+3.5^2
x =
   14.2500
>> y=4*9
y =
   36
>> (x-y)/12
    ans =
      -1.8125
```

يتعامل MATLAB مع جميع المتغيرات كمصفوفات ، ويجب أن يكون اسم المتغير متكوناً من كلمة واحدة لا يفصل بينها مسافة ، فمثلاً : myvariable وليس my variable ، وألا يتعدى ٣١ رمزاً و يبدأ بحرف. يميز MATLAB بين الحروف الصغيرة small case والكبيرة capitals أي أن المتغير A يختلف عن المتغير a. ويجب مراعاة عدم استخدام المتغيرات الخاصة ببرنامج MATLAB مثل :

ans : متغير يرمز للنتائج

pi : القيمة نق

NaN : رقم غير مقبول

inf : مالا نهاية

تتم العمليات الحسابية باستخدام الإشارات التالية :

عملية الجمع	+
عملية الطرح	-
عملية الضرب	*
عملية الأس	^
عملية القسمة	/

يقوم MATLAB بالحسابات مستخدماً ١٥ خانة رقمية ، أي الدقة المضاعفة Double Precision وعادة يتم عرض خمس خانات فقط ، ويمكن عرض كل الخانات إذا تم إدخال الأمر:

```
>> format long
>> ans
ans =
-1.81250000000000
```

للعودة للعرض القصير ندخل *format short* وللعرض بصيغة scientific notation :

```
>> format short e
>> ans
ans =
-1.8125e+000
```

إذا أدخلت عبارة خاطئة فإن MATLAB يرد بعبارة بلون أحمر توضح الخطأ، فمثلاً :

```
>> (x-y/12
??? (x-y/12
Error: Incomplete or misformed expression or statement.
```

من أهم مزايا MATLAB هو أن الأعداد المركبة يتم إدخالها بسهولة كما لو كانت قيماً حقيقية ، ويمكن استخدام الحرف i او j للدلالة على العدد التخيلي. وتتم العمليات الحسابية بالأعداد المركبة بسهولة.

```
>> c=1+2*i
 c =
   1.0000 + 2.0000i
 >> c=1+2*j
 c =
   1.0000 + 2.0000i
>> (1+2i)*(1-2i)
ans =
    5
```

نستطيع استخدام دالة *real* ، ودالة *imag* للتعرف على أجزاء العدد المركب الحقيقي و التخيلي :

```
>> imag(c)
ans =
     2
>> real(c)
ans =
1
```

(١,٤) المتجهات والمصفوفات

المتجهات والمصفوفات هي أساس العمل في بيئة MATLAB . يتم تعريف المتجهات عمودية أو صفية و هي مجموعة أرقام تفصلها فاصلة أو فراغ بين الأقواس المربعة [ ] وقد تمثل السرعة أو القوة أو أي قيمة فيزيائية أخرى. ويتم إدخال المتجه الصفي كالآتي :

```
>> u=[2 3.6 0.5 sqrt(3)]
u =
   2.0000    3.6000    0.5000    1.7321
```

أما المتجه العمودي فنفصل بين عناصره بالفاصلة المنقوطة :

```
>> v=[1;3;7.8;pi]
v =
1.0000
3.0000
7.8000
3.1416
```

المصفوفة matrix هي عبارة عن مجموعة من الأعداد الحقيقية (أو المركبة) عناصرها مرتبة في جدول مستطيل، يسمى كل سطر أفقي من عناصر المصفوفة صفاً (row) ويسمى كل سطر رأسي عموداً (column) فمثلاً المصفوفة A ذات الأبعاد 3×4 تكتب :

```
>> A=[1 2 3 4;6.2 4 5 -3;1/2 1/3 4 -4]
A =
   1.0000    2.0000    3.0000    4.0000
```

```
  6.2000    4.0000    5.0000   -3.0000
  0.5000    0.3333    4.0000   -4.0000
```

وتقوم الفاصلة المنقوطة بمنع ظهور ناتج الأمر في command window بعد تنفيذه، وهذا يفيد في حال المصفوفات الكبيرة.

(١,٤,١) رتبة المصفوفة

لإيجاد رتبة المصفوفة matrix rank نستخدم الأمر *rank* :

```
>> rank(A)
ans =
     3
```

(١,٤,٢) بُعد المصفوفة

أما لمعرفة أبعاد المصفوفة matrix size فندخل *size* :

```
>> size(A)
ans =
     3     4
```

(١,٤,٣) طرق التعامل مع المصفوفة

لرؤية عنصر معين في المصفوفة فعلينا تحديد موقعه بالمصفوفة، فمثلاً إذا أردنا العنصر الواقع في الصف الأول و العمود الثاني:

```
>> A(1,2)
ans =
     2
```

ولاستخراج مصفوفة جزئية من A :

```
>> A(1:2,2:3)
ans =
     2     3
     4     5
```

ولإيجاد الصف الثاني من المصفوفة نكتب :

```
>> A(2,:)
ans =
    6.2000    4.0000    5.0000   -3.0000
```

ولإيجاد العمود الثاني من المصفوفة ندخل :

```
>> A(:,2)
ans =
    2.0000
   4.0000
   0.3333
```

كما يمكن تبديل الصفوف في المصفوفة، فمثلاً لو أردنا تبديل الصف الأول والثالث نكتب :

```
>> A([3,2,1],:)
ans =
    0.5000    0.3333    4.0000   -4.0000
   6.2000    4.0000    5.0000   -3.0000
  1.0000    2.0000    3.0000    4.0000
```

وبنفس الطريقة يتم تبديل العمودين الثاني والثالث، ولكن يتغير وضع الفاصلة :

```
>>A(:,[1,3,2,4])
ans =
    1.0000    3.0000    2.0000    4.0000
    6.2000    5.0000    4.0000   -3.0000
   0.5000    4.0000    0.3333   -4.0000
```

يمكن تبديل الصف أو العمود بأي قيمة جديدة، فمثلا لتغيير قيمة الصف الثاني إلى [2 3 5 6] نقوم بإدخال :

```
>> A(2,:)=[2 3 5 6]
 A =
   1.0000    2.0000    3.0000    4.0000
   2.0000    3.0000    5.0000    6.0000
  0.5000    0.3333    4.0000   -4.0000
```

أما إذا احتجنا إلى تغيير شكل المصفوفة ندخل أمر *reshape* الذي يضم اسم المصفوفة و الشكل الجديد، مثل:

```
>> reshape(A,1,12)
ans =
  Columns 1 through 9
 1.00   2.00   0.500   2.00   3.00   0.333   3.00   5.00  4.00
 Columns 10 through 12
 4.00   6.00  -4.00
```

(١,٤,٤) إنشاء متجهات في MATLAB

وإذا أردنا إدخال متجه أو مصفوفة كبيرة، فإن كتابتها عنصراً بعنصرٍ ليس أمراً عملياً لذلك إن وجدت قاعدة معينة أو نمط متكرر للعناصر نستطيع استخدامها مع إشارة العمود (:) فمثلاً التعبير (1:10) = x يمثل متجهاً صفياً يحتوي على الأرقام الصحيحة من ١ إلى ١٠. ومن أجل الحصول على متتالية حسابية أساسها غير الواحد، يمكننا أن نستخدم إشارة العمود بإدخال القيمة الابتدائية للمتجه، ومقدار الإضافة في كل خطوة، ثم القيمة النهائية. على سبيل المثال نريد كتابة متجه x يحتوي على العناصر (1,1.1,1.2,1.3,...,5.9,6) ندخل:

```
>> x=1:0.1:6
```

وسينتج متجه سطري يحتوي على 51 عنصراً. نستطيع أيضاً استخدام الأمر *linspace* لنفس الغرض، فنحدد البداية، والنهاية وعدد العناصر:

```
>> x=linspace(1,6,51)
```

وهذا يساعد في تقسيم الفترات إلى فترات متساوية، فعلى سبيل المثال الفترة $[0,2\pi]$ تقسم إلى ٢٠ فترة باستخدام الأمر *linspace(0,2*pi,20)* . ولإنشاء متجه صفي مكون من ١٠ عناصر كلها لها القيمة واحد، نستخدم الأمر *ones* مع تحديد حجم المتجه:

```
>> x=ones(1,10)
 x =
  1    1    1    1    1    1    1    1    1    1
```

أو لو احتجنا متجهاً صفرياً فنستخدم الأمر *zeros* :

```
>> y=zeros(1,10)
y =
   0    0    0    0    0    0    0    0    0    0
```

عند إنشاء المصفوفة الصفرية z بالحجم *n x m* نحدد ذلك في الأمر *z =zeros(n,m)*. ولإنشاء مصفوفة w بالحجم *n × m* ، ولكن عناصرها اختيرت عشوائياً ، فنستخدم الأمر *w=rand(n,m)* .

كما يمكن إنشاء مصفوفات أكثر تعقيداً باستخدام مصفوفات أخرى ، فمثلاً العبارتان التاليتان :

```
 >> C=[6.5 4.7 ;.5 4.1];
>> D=[C zeros(size(C)); zeros(size(C)) ones(size(C))]
D =
   6.5000    4.7000         0         0
   0.5000    4.1000         0         0
        0         0    1.0000    1.0000
        0         0    1.0000    1.0000
```

تعطينا الناتج : المصفوفة الجديدة D بضعف حجم المصفوفة C .

وإذا أردنا إنشاء مصفوفة قطرية diagonal matrix بقيم معينة نستخدم *diag* :

```
>> D=diag([1 2 3])
D =
   1    0    0
   0    2    0
   0    0    3
```

الأمر *sum* يقوم بحساب مجموع عناصر المتجه ، فإذا رجعنا للمصفوفة D فإن D(1:k,j) هي العناصر الـ k الأولى في العمود j ، وإذا أردنا حساب مجموع عناصر

العمود الرابع في D نكتب sum(D(1:4,4)) . كما توجد طريقة أخرى للحصول على نفس النتيجة وذلك باستخدام sum(D(:,end)) حيث إن end تدل على العمود الأخير في المصفوفة.

(١,٥) جبر المصفوفات Matrix Algebra

سنتناول بعض الأمور الجبرية المتعلقة بالمصفوفات، ونبين الأوامر التي تنفذها على MATLAB ، وفق التعاريف المعتمدة في علم الجبر الخطي.

(١,٥,١) جمع وطرح المصفوفات

إذا كانت المصفوفتان A و B من النوع n x m فإن مجموعهما الذي يمثل بـ A+B هو مصفوفة من النوع n x m ، عناصرها هي $a_{ij}+b_{ij}$ لكل i=1,2,...n و j=1,2,...m

مثلاً :

```
>> B=[1 3 4;1 2 5;0 1 4]
B =
   1    3    4
   1    2    5
   0    1    4
>> C=[0 3 4;1 2 3;4 5 6]
C =
   0    3    4
   1    2    3
   4    5    6
>> B+C
ans =
   1    6    8
   2    4    8
   4    6   10
```

والفرق بين المصفوفتين :

```
>> B-C
ans =
     1    0    0
```

```
  0    0    2
 -4   -4   -2
```

(١،٥،٢) ضرب المصفوفات

نفرض A مصفوفة من النوع n x m و B مصفوفة من النوع m x p ، تمثل مصفوفة جداء A في B بالرمز A*B ، وهي مصفوفة من النوع n x p وتأخذ عناصرها الشكل

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$ لكل j=1,2,...p و i=1,2,...n

أي يمكن اعتبارها مجموع جداءات عناصر الصف i من A بالعناصر المقابلة من العمود j من B، لذلك يجب تساوي عدد صفوف B مع عدد أعمدة A . وفي حال عدم توفر ذلك فإن البرنامج يقوم بتنبيه المستخدم .

```
>> A
 A =
    1    2    3
    4    5    6
    0    5    7

>> B
B =
   1    6    1    0
   2    8    9    1
   3    7    9    2

>> A*B
 ans =
14     43    46     8
32    106   103    17
31     89   108    19

>> B*A
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

ويمكن حساب أي عبارة رياضية تحتوي على مصفوفات وأعداد حقيقية، مثلاً :

```
>> 2*A-1
ans =
     1    3    5
     7    9   11
    -1    9   13
```

```
>>5*(A*A)^3
ans =
     1101285   3119175   4131720
     2591400   7339635   9722220
     2431300   6886200   9121595
```

ولكن لإجراء عمليات جبرية على مستوى العناصر نضع نقطة قبل العملية مثل

*. أو /. أو ^.

```
>>5>>5*(A.*A).^3
ans=
        5       320       3645
    20480     78125     233280
        0     78125     588245
```

وتختلف النتيجة ما بين العملية * ونفس العملية مرفوقة بالنقطة * . لأن العملية الأولى هي عملية ضرب مصفوفتين بينما العملية الثانية هي عملية ضرب العنصر بالعنصر، كما تبرهن الخطوات التالية :

```
>> a=[1 2;3 4]
a =
   1    2
   3    4
>> b=[5 6;7 9]
 b =
      5    6
      7    9
 >> a*b
 ans =
      19   24
```

```
     43   54
>> a.*b
ans =
      5   12
     21   36


>> a.^b
ans =
        1        64
        2187     262144
```

في العملية الأخيرة كل عنصر من *a* رفع للقوة المقابلة في b.

(١,٥,٣) مصفوفة الوحدة و معكوس المصفوفة

المصفوفة المربعة ببعد $n \times n$ التي تحتوي على أصفار ما عدا على القطر تأخذ القيم ١ ، تُدعى مصفوفة الوحدة identity matrix وتنتج بالأمر *eye(n)* ، فمصفوفة الوحدة من الرتبة الثالثة هي :

```
>> eye(3)
   ans =
          1    0    0
          0    1    0
          0    0    1
```

وإذا كان لدينا مصفوفة مربعة A فيمكن حساب محدد المصفوفة matrix determinant بإدخال الأمر *det(A)* :

```
>> A
A =
   1    2    3
   4    5    6
   0    5    7
>> det(A)
 ans =
      9
```

عندما يكون المحدد للمصفوفة المربعة غير صفري، فيمكن حساب معكوس المصفوفة matrix inverse بإدخال A^{-1} أو inv(A) :

```
>> inv(A)
ans =
0.5556   0.1111  -0.3333
-3.1111   0.7778   0.6667
2.2222  -0.5556  -0.3333
```

بحيث يكون $I = A\,A^{-1} = A^{-1}A$ و I مصفوفة الوحدة بحجم A ، وبذلك تكون المصفوفة غير شاذة nonsingular . ويمكن التأكد بحساب $A\,A^{-1}$:

```
>> inv(A)*A
ans =
   1.0000       0          0.0000
      0       1.0000       0
     0        0           1.0000
```

إذا كانت المصفوفة B غير مربعة $m \times n$ بحيث $m > n$ فيمكن لـ MATLAB حساب شبه المعكوس pseudo- inverse بدالة *pinv* ، ويمكن الاطلاع على التعريف الرياضي لشبه المعكوس بالأمر *help pinv*.

```
>> B=[1 2 3;4 5 9;7 11 18;-2 3 1;7 1 8];
>> pinv(B)
ans =
  -0.0080   0.0031  -0.0210  -0.0663   0.0966
   0.0117   0.0092   0.0442   0.0639  -0.0805
   0.0036   0.0124   0.0232  -0.0023   0.0162
>>help pinv
```

MATLAB Function Reference pinv

Moore-Penrose pseudo inverse of a matrix

B = pinv(A)

B = pinv(A,tol)

Definition

The Moore-Penrose pseudo inverse is a matrix B of the same dimensions as

A' satisfying four conditions:
*A*B*A = A*
*B*A*B = B*
*A*B is Hermitian*
*B*A is Hermitian*
If A is square and not singular, then pinv(A) is an expensive way to compute inv(A). If A is not square, or is square and singular, then inv(A) does not exist. In these cases, pinv(A) has some of, but not all, the properties of inv(A).

(١,٥,٤) القيم الذاتية للمصفوفة

الأمـــر poly يعـــــطي متجهـــاً يحتـــــــوي علـــى معـــاملات المعادلـــة المميزة characteristic equation للمصفوفة A ، حيث إن المعادلة المميزة للمصفوفة هي det(λ I-A)=0. ويمكن حساب القيم الذاتية Eigenvalues للمصفوفة A عن طريق تطبيق الأمر roots على المعادلة المميزة للمصفوفة. كما يوجد في MATLAB دالة جاهزة لحساب القيم الذاتية للمصفوفة تُدعى eig .

نطبق على مثالنا الســـابق ونلاحظ تطابق الطريقـــتين، حيث إن جذور المعادلة المميـــزة لـ A ‏($\lambda^3-13\lambda^2+9\lambda-9=0$) تعطي القيم الذاتية للمصفوفة .

```
>> p=round(poly(A))
p =
   1  -13    9   -9

>> roots(p)
ans =
          12.3292
           0.3354 + 0.7858i
           0.3354 - 0.7858i
>> eig(A)
ans =
           0.3354 + 0.7858i
           0.3354 - 0.7858i
          12.3292
```

(١,٥,٥) منقول المصفوفة

منقول المصفوفة matrix transpose $A = (a_{ij})$ من النوع n×m هو المصفوفة $A^t = (a_{ji})$ من النوع m×n بحيث تصبح الصفوف في A أعمدة في A^t والعكس. ولحساب المنقول نستخدم علامة الاقتباس المفردة (') وتقوم هذه العملية بقلب عناصر المصفوفة وفق قطرها الرئيس على النحو التالي :

```
>> A'
ans =
      1    4    0
      2    5    5
      3    6    7
```

```
>> B=[ 1 2 ;3 4;5 6]'
ans =
      1    3    5
      2    4    6
```

حتى في حال مصفوفة غير مربعة

تسمى المصفوفة المربعة متناظرة symmetric matrix إذا كان $G = G^t$ مثل :

```
>> G
G =
   6    4   -3
   4   -2    0
  -3    0    1
>> G'
ans =
      6    4   -3
      4   -2    0
     -3    0    1
```

إذا كانت المصفوفة تحوي أعداداً مركبة فإن (') تعطي المنقول المرافق المركب complex conjugate transpose :

```
>> a=[1+2*i 3+5*i;4+2*i 3+4*i]
 a =
1.0000 + 2.0000i   3.0000 + 5.0000i
4.0000 + 2.0000i   3.0000 + 4.0000i
```

```
>> a'
1.0000 - 2.0000i   4.0000 - 2.0000i
3.0000 - 5.0000i   3.0000 - 4.0000i
```

لحساب المنقول المركب من غير المرافق complex transpose نسبق العلامة بنقطة أي (.') :

```
>> a.'
    1.0000 + 2.0000i   4.0000 + 2.0000i
    3.0000 + 5.0000i   3.0000 + 4.0000i
```

(١,٥,٦) نظيم المتجه و نظيم المصفوفة

يحسب MATLAB النظيم *norm* للمتجهات أو للمصفوفات على حسب نوع النظيم بالأوامر المذكورة في *help* :

```
For matrices...
    NORM(X,2) is the same as NORM(X).
    NORM(X,1) is the 1-norm of X, the largest column sum,
                = max(sum(abs(X))).
    NORM(X,inf) is the infinity norm of X, the largest row sum,
                = max(sum(abs(X'))).
    NORM(X,'fro') is the Frobenius norm, sqrt(sum(diag(X'*X))).
For vectors...
    NORM(V,P) = sum(abs(V).^P)^(1/P).
    NORM(V) = norm(V,2).
    NORM(V,inf) = max(abs(V)).
```

فمثلاً النظيم ℓ_∞ norm المُعرف $\|x\|_\infty = \max_i |x_i|$ للمتجه $x = [-1\ 1\ -3]$ يحسب بالأمر التالي :

```
norm(x,inf)
ans =
    3
```

(١,٥,٧) العدد الشرطي للمصفوفة

يمكن إيجاد العدد الشرطي لمصفوفة باستخدام دالة *cond* وذلك حسب التعريف $cond\,(A) = \|A\|\,\|A^{-1}\|$ ، فمثلاً إذا تم اختيار النظيم الإقليدي ℓ_2 norm (p=2) على المتجه $x = (x_1, x_2, ... x_n)^t$ فإن نظيم المصفوفة :

$$\|A\|_2 = \max \|Ax\|_2 \quad s.t. \quad \|x\|_2 = \left\{ \sum_{i=1}^{n} x_i^2 \right\}^{1/2}$$

يمكن تحديده بالأمر *cond* (A, 2) . وتُعد المصفوفة حسنة الشروط well-conditioned كلما اقترب العدد الشرطي من الواحد، وتدعى سيئة الشروط ill-conditioned عندما يكون العدد الشرطي أكبر من الواحد بصورة ملحوظة.

```
>> A=[3 2 -1;-1 3 2;1 -1 -1 ] ;
>> cond(A)
  63.0547
```

(١,٦) الدوال المخزّنة على MATLAB

توجد مجموعةٌ كبيرةٌ من الدوال مخزنة ذاتياً في MATLAB، ويمكن استحضار الدوال بطباعة اسم الدالة والعوامل المرتبطة بالدالة. وقد تعطي هذه الدوال ناتجاً أو أكثر على حسب نوعها. الجدول رقم (١.١) يعرض بعض هذه الدوال ، حيث يبين اسم الدالة ووظيفتها، و مثالاً على طريقة استعمالها. ونلاحظ استخدام الحروف الصغيرة في جميع أسماء الدوال. بعض الدوال تحتاج إلى أكثر من عامل إدخال لتحصل على النتائج المرجوة، مثل *bessel(n,x)* وهي دالة البسل بدرجة n للمتغير x. وفي المقابل هناك الدالة *[y,j]=sort(x)* وهي مثال على دالة الترتيب التي تعطي قيمتي إخراج، y هي المصفوفة المرتبة وj هي المصفوفة التي تحتوي على معاملات العناصر لهذا الترتيب.

الجدول رقم (١,١) . دوال جاهزة في MATLAB.

الدالة	وظيفتها	مثال
exp(x)	e^x	3*exp(4)
floor(x)	تقريب باتجاه ∞ -	floor(3.6)
ceil(x)	تقريب باتجاه ∞+	ceil(4.9)
gcd(x)	القاسم المشترك الأكبر	gcd(6,8)
lcm(x)	المضاعف المشترك الأصغر	lcm(12,10)
sqrt(x)	الجذر التربيعي	sqrt(5.8+2)
max(v)	القيمة العظمى	max([2,4,7])
min(v)	القيمة الصغرى	min([2,4,7])
imag(z)	الجزء التخيلي من العدد المركب	imag(4+6*i)
real(z)	الجزء الحقيقي من العدد المركب	real(4+6*i)
fix(x)	تقريب باتجاه الصفر	1-fix(5.9)
round(x)	تقريب باتجاه أقرب رقم صحيح	3*round(6.8)
sum(v)	حاصل جمع العناصر	sum([2,3,6)]
abs(x)	القيمة المطلقة	3*abs(-4.5)

نعرض بعض الدوال المثلثية ومعكوس كلٍ منها في الجدول رقم (١,٢).

الجدول رقم (١,٢) . الدوال المثلثية في MATLAB.

sin(x)	دالة الجيب المثلثية
cos(x)	دالة الجيب التمام المثلثية
tan(x)	دالة الظل المثلثية
sec(x)	دالة القاطع المثلثي
csc(x)	دالة القاطع التمام المثلثي

تابع الجدول رقم (١،٢) .

cot(x)	دالة الظل التمام المثلثي
acos(x)	معكوس cos
asin(x)	معكوس sin
acot(x)	معكوس cot
atan(x)	معكوس tan

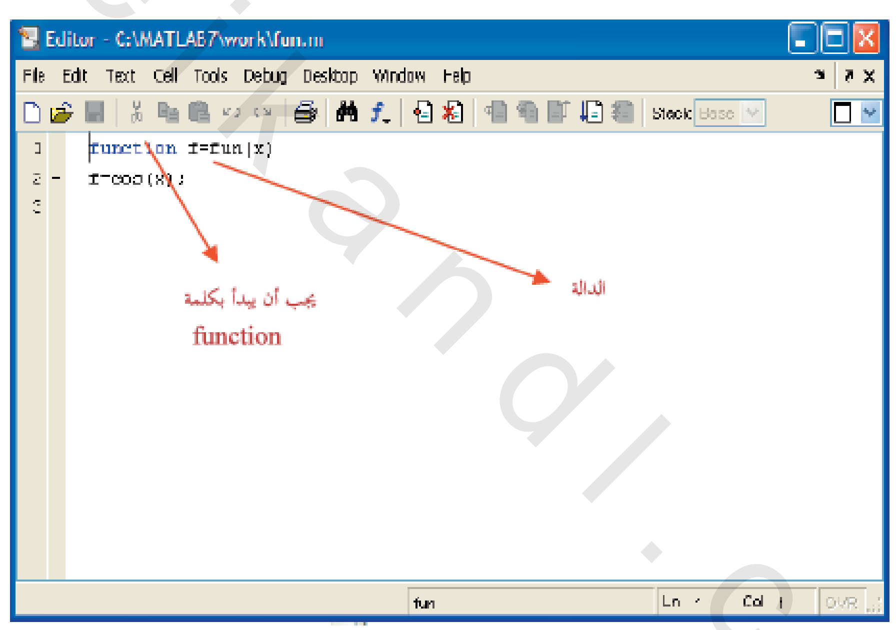
(١،٧) تعريف دوال في MATLAB

(١،٧،١) تعريف الدالة في ملف m-file

يستخدم MATLAB نوعين من الملفات التي تدعى m-files. النوع الأول هو ملف لكتابة البرامج ويخزن باسم معين للبرنامج ويمكن استحضار البرنامج وتشغيله بطباعة اسم الملف في نافذة الأوامر أو في أي برنامج آخر. النوع الثاني هو ملف لتعريف الدوال function m-file ويخزّن باسم الدالة المراد تعريفها ، ويتم استخدام الدالة عند طباعة اسم الدالة في نافذة الأوامر مثل أي دالة جاهزة في MATLAB. ويمكن للمستخدم تعريف دوال خاصة وفق صيغة معينة ، وبذلك يتمكّن من القيام بتطبيقات عديدة. الصيغة العامة هي :

function output=function_name(input)
function body

وسوف يتم حفظ الملف بنوع fun. m ويجب تعريف الدالة في أول سطر بكلمة *function* ثم يلي ذلك أسطر تحتوي على حسابات الدالة. مع مراعاة احتواء الدالة على عامل الإدخال input عند طباعة اسمها ، وستظهر النتيجة في عامل الإخراج output الشكل رقم (١,٤).



الشكل رقم (١,٤). نافذة تعريف الدوال Function m-file.

مثال رقم (١,٢)

عَرِّف الدالة البسيطة $f(x)=\left(\frac{x}{2}\right)^3-x+\sin\left(\frac{\pi x}{2}\right)$ في ملف يحمل اسم *fun*

واحسب قيمتها عند (2.5).

الحل :

نعرف الملف كالتالي :

```
function p=fun(x)
% A simple function definition
x=x/2;
p=x^3-2*x+sin(pi*x);
```

ويمكن حساب قيمة الدالة عند (2.5) بطباعة *y = fun(2.5)* . أو باستخدام الأمر feval('fun',2.5) .

مثال رقم (١,٣)

عرف الدالة في متغيرين $f(x,y)=x+ye^{x^2+y^2}$ واحسب قيمتها عند (1.2).

الحل :

نستطيع تعريفها في ملف fun2 كالآتي :

```
function z=fun2(x,y)
z=x+y.*exp(x.^2+y.^2);
```

ومن ثم تُستخدم الدالة لإيجاد قيمتها عند النقطة المطلوبة ، بالأمر fun2(1.2) لتعطي القيمة 297.8263 .

كما يمكن تطبيق الدوال الجاهزة في MATLAB على الدوال المعرفة من قبل المستخدم ، مثل دالة *fzero* التي تستخدم لإيجاد جذور دالة بطرق عددية. فلإيجاد جذر *fun* في الفترة ما بين 2 و4 نكتب *fzero('fun',[2,4])* (ملحوظة : كتابة اسم الدالة تتم

داخل علامات اقتباس) وسيُظهر MATLAB الحل في شاشة الأوامر 3.1264 .

(١,٧,٢) أوامر تعريف الدوال

طريقة أخرى لتعريف دالة في سطر واحد هي استخدام الأمر *inline*. و يُحدد اسم الدالة بين علامات الاقتباس والمتغيرات مثل :

```
>> g=inline('exp(-x.^2)','x')
g =
   Inline function:
   g(x) = exp(-x.^2)
```

ثم يمكن حساب قيم الدالة بسهولة عند أي ثابت بالأمر *feval* :

```
>> feval(g,0)
ans =
    1.0000
```

أو مباشرة عند أي متجه :

```
>> x=0:3
x =
    0    1    2    3
>> g(x)
ans =
   1.0000    0.3679    0.0183    0.0001
```

كما يمكن استخدام إشارة @ لتعريف الدالة مثل :

```
>> fh = @(x,y)y*sin(x)+x*cos(y)
fh =
   @(x,y)y*sin(x)+x*cos(y)
>> fh(pi,2*pi)
ans =
   3.1416
```

حساب الدالة عند (π,2 π)

(١,٨) الإدخال والإخراج في MATLAB

(١,٨,١) أوامر الإخراج

أبسط طريقة في الإخراج أو عرض البيانات على MATLAB هي عدم كتابة الفاصلة المنقوطة في نهاية عبارة التعريف، ولكن هذه الطريقة لا تظهر النتائج بطريقة مرتبة، فالأفضل استخدام أوامر خاصة بالإخراج، مثل: الأمر disp الذي يعرض النتائج و العبارات النصية.

لعرض محتويات مصفوفة نستخدم disp(A) و لعرض عبارة نص يجب كتابتها ضمن إشارة الاقتباس ' ' :

```
>> disp('this is text')
this is text
```

ويمكن عرض عبارة تحتوي على نص و قيم رقمية وذلك بفصلها عن بعضها بالفاصلة، وتكون داخل أقواس مربعة ويجب استخدام دالة *num2str* لتحويل الأرقام إلى نص:

```
>> x
x =
  10.8000
>> t
t =
    3
>> disp(['this is the value   ',num2str(x),'    at the time    ',num2str(t)])
this is the value   10.8    at the time    3
```

أما للعرض بشكل منسق، فهناك الأمر الأكثر مرونة وهو *fprintf* ، ويمكن العرض على الشاشة أو على ملف. ويأخذ الصيغة التالية بعد تحديد المتغيرات وأنواعها وعدد خانات العرض مسبوقة بعلامة ٪ :

```
>> fprintf('filename','format string',list);
```

```
>> fprintf('x=%8.2f t=%8.6f ',x,t)
x=   10.80 t=3.000000
```

(١,٨,٢) أوامر الإدخال

أما بالنسبة لإدخال نص أو بيانات من لوحة المفاتيح، فنستعمل دالة *input*. وهي تأخذ الصيغة الأولى في حال إدخال قيمة رقمية، و تأخذ الشكل الثاني إذا أردنا إدخال نص. ستظهر على الشاشة كلمة *text* وينتظر البرنامج من المستخدم إدخال القيمة المناسبة باستخدام لوحة المفاتيح:

```
>> x=input('text');
>> x=input('text','s');
```

لإدخال بيانات كثيرة يفضّل استخدام الأمر *load* الذي يمكّن المستخدم من إدخال بيانات تم حفظها في ملف، ويكتب اسمه بعد الأمر، أي *load filename*. عند وجود البيانات في ملف يمكن قراءة البيانات بعد فتح الملف بدالة *fopen* عن طريق أحد الأوامر *fscanf*، *fget1* أو *fread* وكل من هذه الدوال لها صيغة معينة، وتعتمد على نوع الملف ونوع البيانات (*binary*، *text*، ...). ويجب إقفال الملف عند الانتهاء من قراءة البيانات بالأمر *fclose*. لمزيد من التفاصيل عن دوال الإدخال نقترح اللجوء إلى *help*.

(١,٩) الرسم على MATLAB

من أهم مميزات MATLAB القدرة على رسم المنحنيات ثنائية و ثلاثية الأبعاد 3D graphics والتي يصعب رسمها ببقية لغات البرمجة. ويقدم MATLAB

وسائل مساعدة للتحكم بالرسوم وتعديلها، سواء من ناحية تحديد المحاور أو تغيير نمط ولون خط الرسم أو تحريك الشكل. يوفر MATLAB دوال عديدة للرسم ويمكن استخدامها مباشرة على الرسم أو كتابتها على شكل أوامر في نافذة الأوامر.

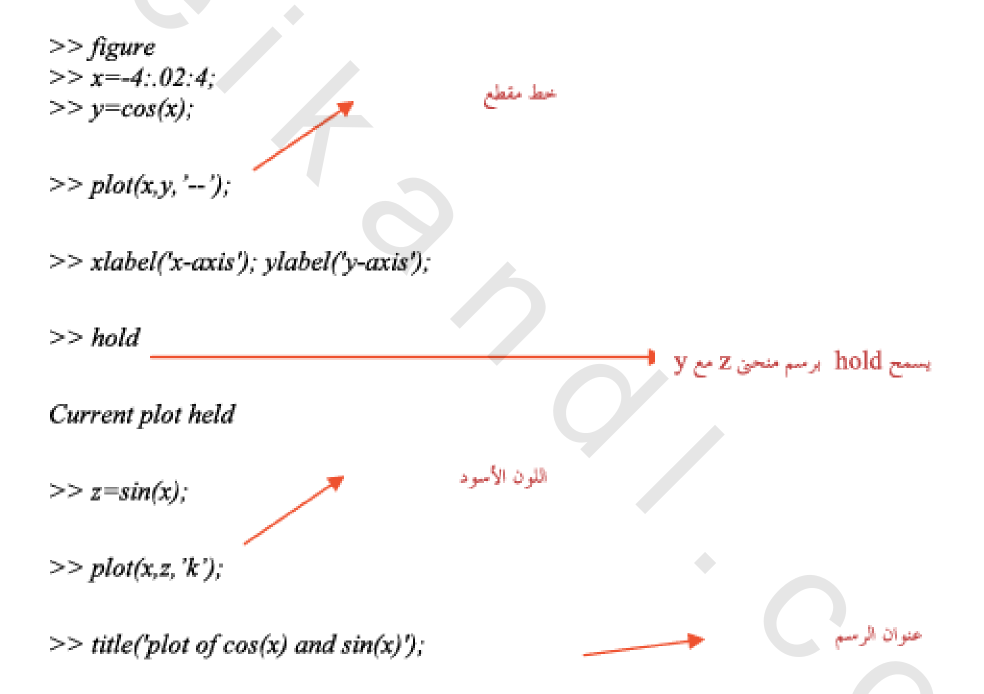
(١,٩,١) الرسم في بعدين

الأمر الأساسي في الرسم في بعدين هو *plot(x,y)* ويقوم هذا الامر بفتح شاشة خاصة بالرسم تسمى figure file و يتم رسم المتجه x في محور السينات والمتجه y في محورالصادات. كما توجد دوال أخرى مثل:

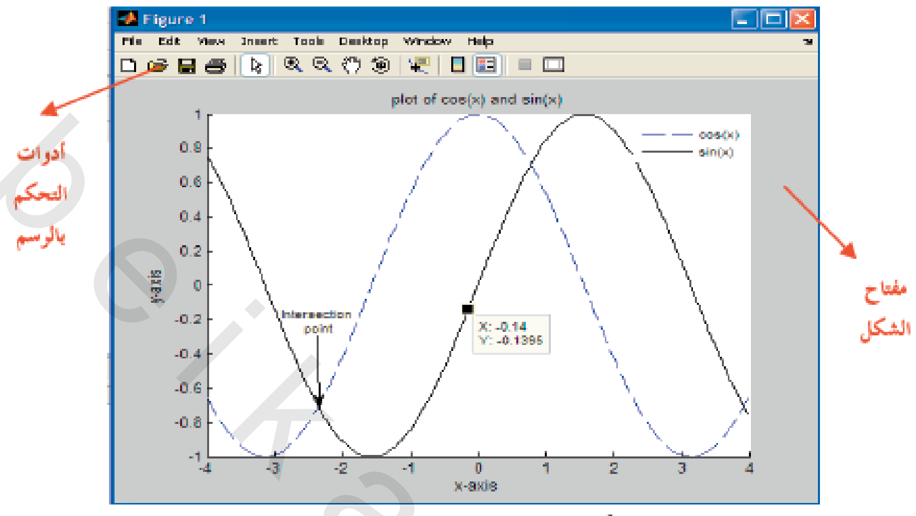
- *hold*: تقدم هذه الدالة إمكانية رسم أكثر من منحنى على نفس الرسم، حيث يتم تفعيلها بـ *hold on* و هذا بعد إنشاء الرسم الأول، ثم يتم رسم المنحنى التالي، ويمكن إيقافها بـ *hold off* .

- *axis*: تقوم بتحديد المحاور مع العلم بأن MATLAB يقوم بذلك بشكل ذاتي ولكن هذه الدالة تستخدم لإظهار أفضل شكل من تكبير أو تصغير أو حتى إظهار جزء من الرسم.
- *title* : تقوم بإدراج عنوان في أعلى الرسم.
- *xlabel* : تستخدم لتحديد تسمية المحور الأفقي للرسم.
- *ylabel* : تستخدم لتحديد تسمية المحور العمودي للرسم.
- *figure* : تستخدم لفتح نافذة للرسم جديدة للعرض.
- *fplot* : تستخدم لرسم دالة معرّفة مسبقا بقيم مختلفة.
- *Plot(x1,y1,x2,y2,x3,y3,...)* : تستخدم لرسم عدة منحنيات على نفس المحاور، باعتبار الأزواج المرتبة (x1,y1), (x2,y2), (x3,y3)...

وهناك وسائل تحكم عديدة على شاشة الرسم، مثل تغيير طريقة الرسم من خط مستقيم إلى خط مقطّع، أوخط يحتوي على رمز توضيح، ويمكن تغيير سُمك الخط ولونه من الجدول رقم (١،٣) مع مراعاة كتابتها بين علامات اقتباس. يستطيع المستخدم إضافة نصّ في أي مكان على الرسم، وكذلك إضافة مفتاح للتمييز بين المنحنيات.



يظهر الرسم الموضح في الشكل رقم (١،٥) على نافذة مرقمة خاصة بالرسم تسمى *Figure*. ويتم التحكم بالخطوط والألوان عن طريق رموز، أدرجنا بعضها في الجدول رقم (١،٣)، وتكتب داخل أقواس دالة *plot*، ويمكن كتابة أكثر من رمز للحصول على الرسم المناسب مثل *plot(x,y,':*r')* التي ترسم المنحنى بخط أحمر مقطع تتخلله نجوم. كما يمكن التحكم بالشكل من خلال شريط الأوامر بالنافذة .



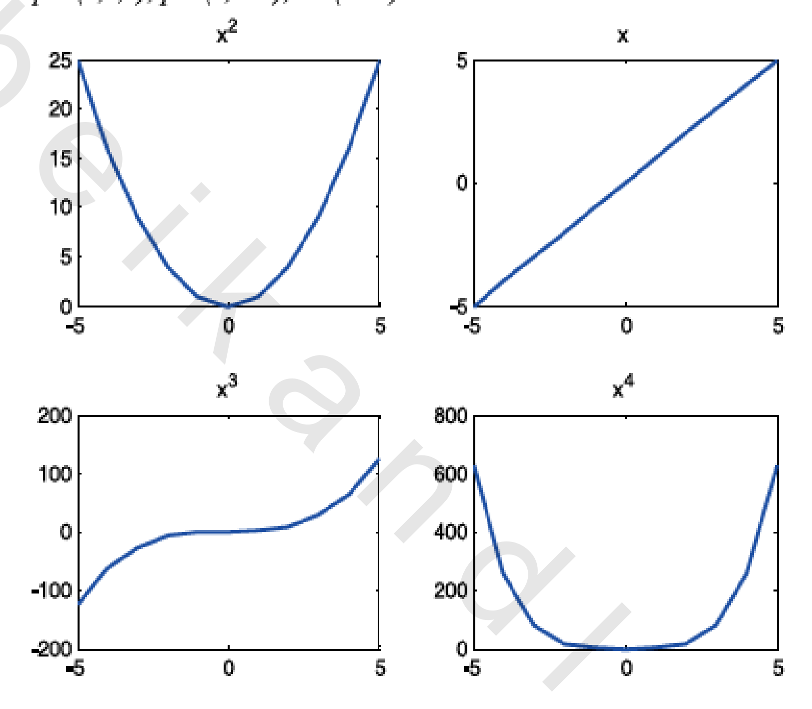
الشكل رقم (١,٥). نافذة الرسم Figure.

الجدول رقم (١,٣). أدوات التحكم بالرسم.

اللون	أحمر	أخضر	أزرق	سماوي	أرجواني	أصفر	أسود	أبيض
الرمز	r	g	b	c	m	y	k	w
نوع الخط	: مقطع بنقاط	– مقطع بواصلة	-. مقطع بنقطة و واصلة	* خط تتخلله نجوم	+ خط يتخلله +	x خط يتخلله x	. خط يتخلله .	بدون رمز خط مستقيم

ولرسم الدوال x^4 ، x^3 ، x^2 ، x مستخدماً الدالة *subplot* التي تمكننا من تجزيء العرض إلى عدد من الخلايا ورسم كل دالة في خلية كما هو موضح في الشكل رقم (١,٦).

```
>> x=-5:5;
>>subplot(2,2,1); plot(x,x.^2);title('x^2')
>> subplot(2,2,3); plot(x,x.^3);title('x^3')
>> subplot(2,2,2); plot(x,x);title('x')
>> subplot(2,2,4); plot(x,x.^4);title('x^4')
```



الشكل رقم (١،٦). مثال على الأمر subplot.

مثال رقم (١،٤)

لرسم دالة *f(x)* غير متصلة ومعرفة كالآتي :

$$f(x)=\begin{cases}\tan x & -\pi/4 \le x \le \pi/4 \\ \sin(x) & \pi/4 \le x \le \pi/2 \\ e^{x} & \pi/2 \le x \le 3\end{cases}$$

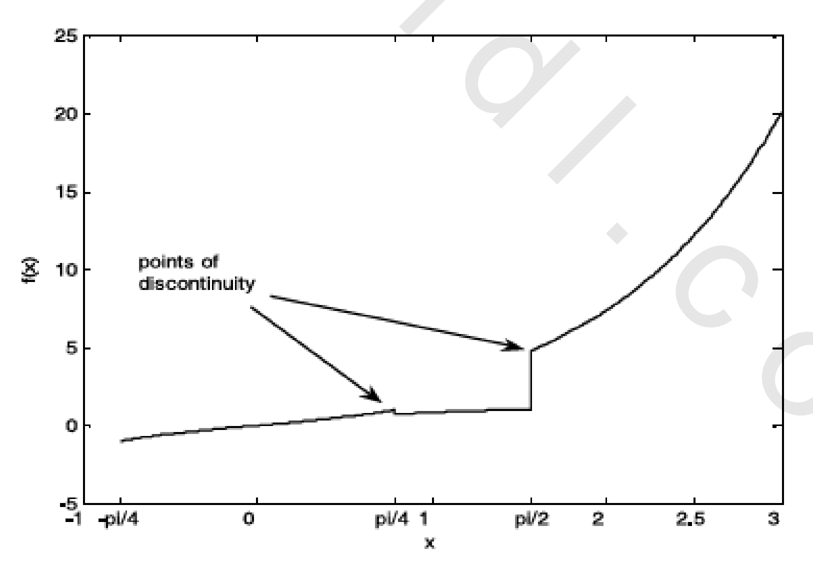
نُدخل كل جزء من مجال الدالة بمتجهات x1 ,x2 ,x3 وكل جزء من الدالة بمتجهات y1, y2, y3 :

```
>> x1=-pi/4:pi/200:pi/4;
>> y1=tan(x1);
>> x2=pi/4:pi/200:pi/2;
>> y2=sin(x2);
>> x3=linspace(pi/2,3);
>> y3=exp(x3);
```

ثم نضم متجهات المجال في متجه واحد x و بالمثل ننشئ متجه y :

```
>> x=[x1 x2 x3];
>> y=[y1 y2 y3];
>> plot(x,y)
```

ونحصل على رسم الدالة *f(x)* الذي يتضح عليه نقاط عدم الاتصال كما في الشكل رقم (١,٧).



الشكل رقم (١,٧). رسم الدالة *f(x)* .

هناك أنواع مختلفة من الرسومات، ومن أهمها :

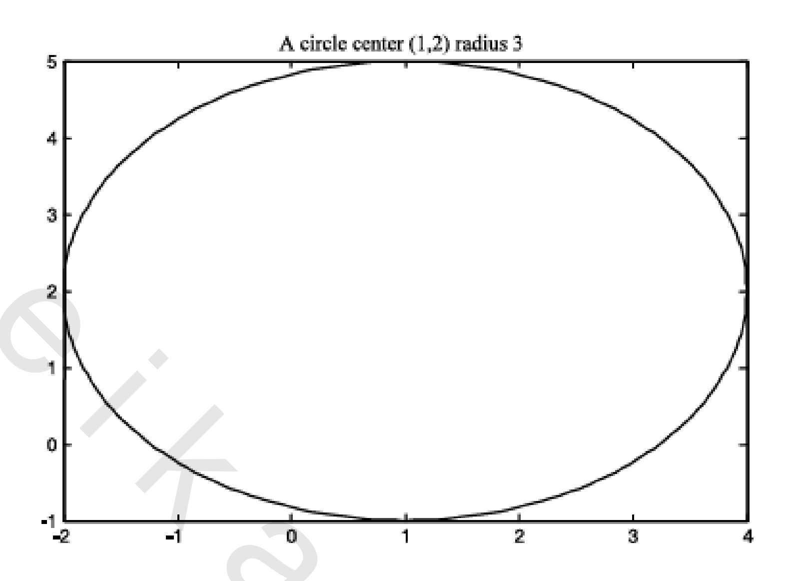
• إذا كانت الدوال معرّفة بمنحنيات وسيطية parametric curves فترسم بنفس الطريقة ، فمثلاً لرسم دائرة بمركز (1,2) ونصف قطر ٣ معرّفة بـ $x=1+2cost$ و $y=1+2sint$ على الفترة $0 \le t \le 2\pi$ نُدخل الأوامر التالية لنحصل على الشكل رقم (١,٨) .

```
>> t=0:2*pi/200:2*pi;
>> plot(1+3*cos(t),2+3*sin(t))
```

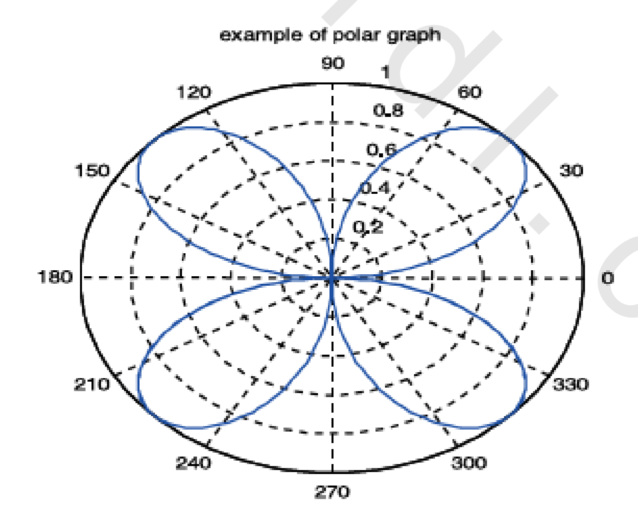
• الدالة *polar(theta, r)* تنتج رسماً بمحاور قطبية بالزوايا مدخلة في المتجه theta و القيم مدخلة في المتجه r ، فمثلاً الأمر *polar(x,sin(2*x))* يعطي الشكل رقم (١,٩) .

• نرغب أحيانا في تمثيل البيانات على شكل Bar chart أو على شكل Pie chart أو نقاط متفرقة Scatter points ويقدم MATLAB دوال جاهزة لذلك. على سبيل المثال pie(a,b) حيث a متجه يمثّل نسبة درجات الطلاب في مادة معينة ، و b طريقة التجزيء (هنا اخترنا أكبر قطعة تمثل التقدير C) تعطي الشكل رقم (١,١٠) .

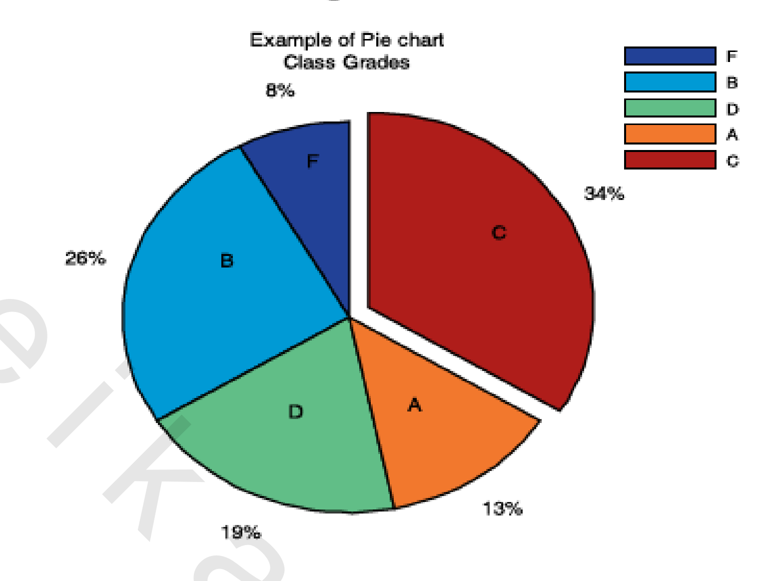
• الأمر bar(x) ينتج الرسم في الشكل رقم (١,١١) .



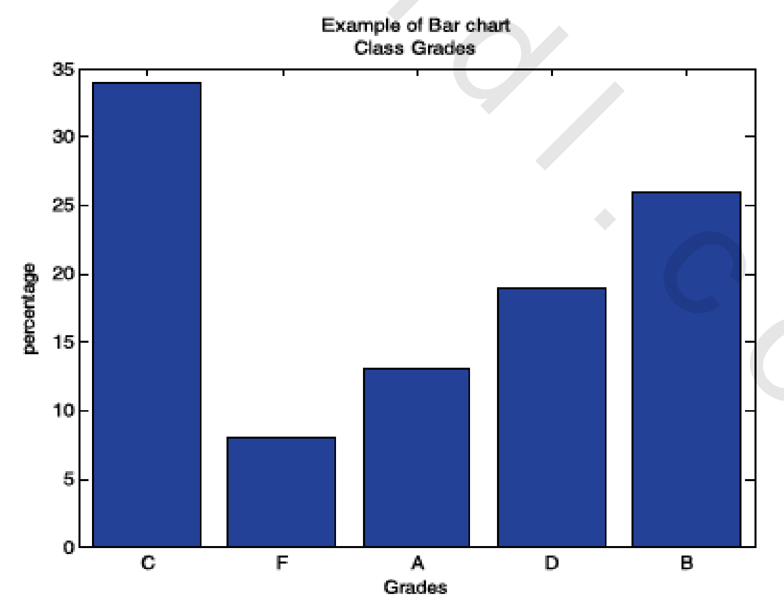
الشكل رقم (١,٨). رسم دائرة بنصف قطر ٣ ومركز (1,2).



الشكل رقم (١,٩). مثال لرسم قطبي.



الشكل رقم (١,١٠). مثال لرسم Bar chart.



الشكل رقم (١,١١). مثال لرسم Bar chart .

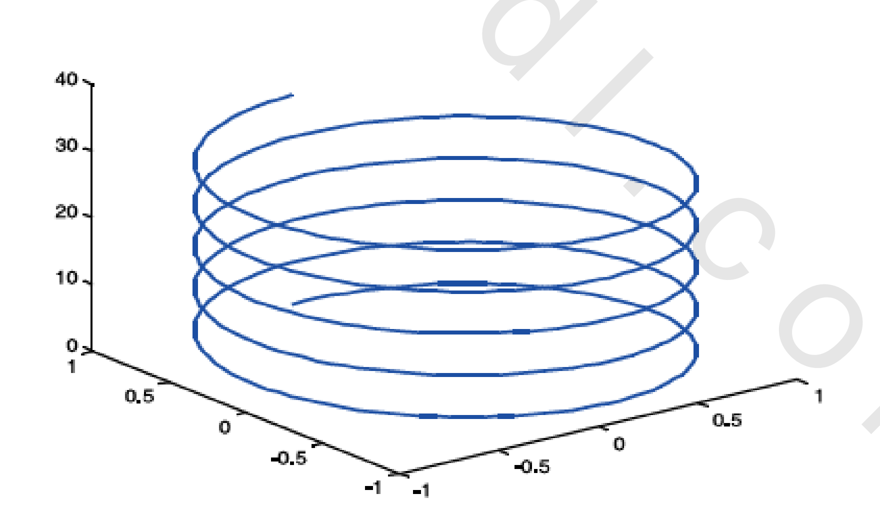
(١,٩,٢) الرسم في الأبعاد الثلاثية

يمكّن MATLAB المستخدم من العرض البياني في ثلاثي الأبعاد على شكل خطوط ذات ثلاثة أبعاد أو مسطحات متنوعة، وسنعرض في هذا الجزء نبذة عن بعض الدوال التي تتحكم في هذا النوع من الرسوم، وللتعرف أكثر على ما يمكن استخدامه من أوامر أخرى ننصح بالرجوع لأمر المساعدة *help graphics*.

الدالة *plot3(x,y,z)* ترسم رسماً ثلاثي الأبعاد، وقد استخدمنا الأوامر:

```
t = 0:pi/50:10*pi
plot3(sin(t),cos(t),t )
```

والناتج هو الرسم الحلزوني في الشكل رقم (١,١٢).



الشكل رقم (١,١٢). مثال لرسم بالأمر plot 3 .

كما توجد دوال أخرى للرسم مثل *meshgrid, mesh, surf, contour, contour3* ، ولكن في البداية نحدد مجال الدالة بتعريف مصفوفتين للإحداثيات x و y ونستخدم دالة *meshgrid* :

```
>> [x y]=meshgrid(-8:0.5:8);
```

ثم نعرّف الدالة z ، ونستخدم الرقم $eps = 2.2x10^{-16}$ لتفادي القسمة على صفر.

```
>> r=sqrt(x.^2+y.^2)+eps;
>> z=sin(r)./r;
```

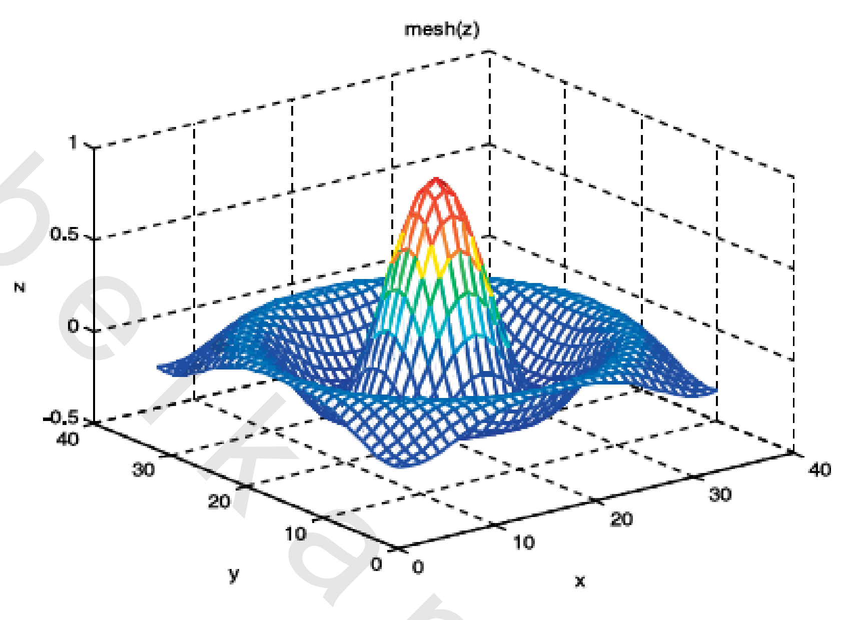
وهنا بعض الأمثلة لرسم القبعة المكسيكية باستخدام دوال مختلفة :

١- الأمر *mesh* : ينتج الرسم على شكل شبكة كما في الشكل رقم (١,١٣)، بحيث تأتي نقاط تقاطع الأعمدة والصفوف في الشبكة من المصفوفات المحددة في *meshgrid* .

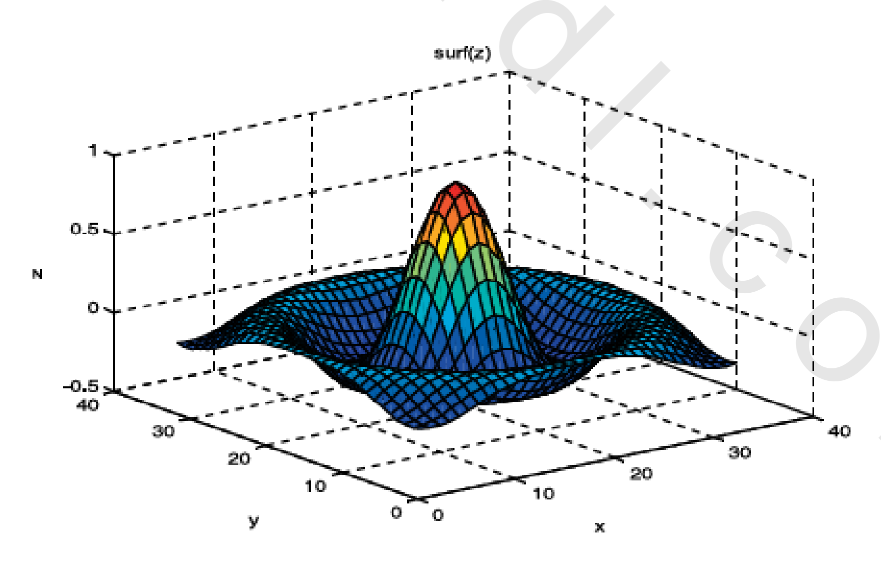
٢- لرسم مسطح نستخدم الدالة *surf* ، وينتج الشكل رقم (١,١٤).

٣- يمكن رسم خطوط محددة للرسم بدالة *contour3* المبينة بالشكل رقم (١,١٥).

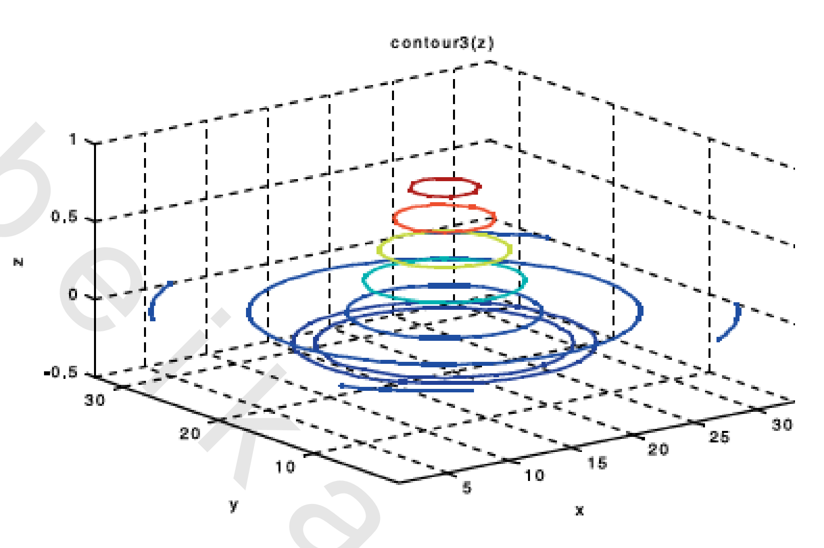
٤- يمكن رسم الخطوط تحت المسطح أو الرسم الشبكي بدالتي *surfc* و *meshc* مثل الشكل رقم (١,١٦).



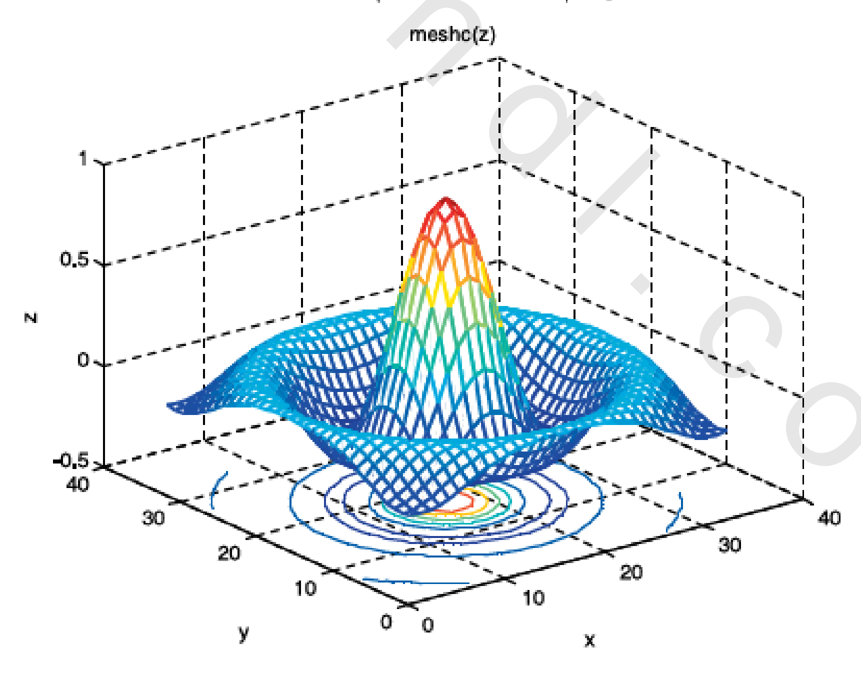
الشكل رقم (١,١٣). مثال لرسم بالأمر mesh.



الشكل رقم (١,١٤). مثال لرسم بالأمر surf .



الشكل رقم (١,١٥). مثال لرسم بالأمر contour3 .



الشكل رقم (١,١٦). مثال لرسم بالأمر meshc .

(١,١٠) العلاقات وعمليات المنطق الرياضي في MATLAB

بالإضافة للعمليات الرياضية التقليدية التي يمكن القيام بحسابها في MATLAB فإنه يقوم بإجراء عمليات وعلاقات في المنطق الرياضي Logical Operators. ويقدم MATLAB مبدأ جديداً وهو المتجه المنطقي logical vector وهذه خاصية قوية بـ MATLAB ولا توجد في أي لغة أخرى. الهدف من عمليات المنطق الرياضي هو الرد على الأسئلة الصائبة و الخاطئة، و يقوم بهذه العملية MATLAB ويخزّن النتائج في المتجه المنطقي الذي يحتوي على القيمة 1 للصائب و القيمة 0 للخاطئ. أما أدوات الربط في المنطق الرياضي على MATLAB فهي موضحة في الجدول رقم (١,٤). نستخدم أدوات الربط للمقارنة بين متجهين بنفس الحجم، أو لمقارنة عدد ثابت بمتجه. ونقارن الأعداد ببعضها وتكون الإجابة بصائب (1) أو خاطئ (0).

أكثر استخدام لهذه المقارنات هو في مجال البرمجة، لأن استخدامها يعطي البرامج سرعة ويجعل المهمة التي تحتاج أسطراً عديدة لتنفيذها تُنفذ بأمر واحد من MATLAB، وسوف نعرض أمثلة على البرمجة في الجزء القادم.

الجدول رقم (١,٤). أدوات المنطق الرياضي.

أقل	<
أقل من أو يساوي	<=
اكبر من أو يساوي	>=
يساوي	==
أكبر	>
عدم المساواة	~=
النفي	~
وَ	&
أو	|

مثال رقم (٥,١)

نفرض أن لدينا متجهين A وB :

```
» A=1:11, B=16-A,

A =
    1   2   3   4   5   6   7   8   9   10   11

B =
    15   14   13   12   11   10   9   8   7   6
```

إذا قمنا بمقارنة المتجه A وعدد ثابت 5 :

```
» r=A>5
r =
    0   0   0   0   0   1   1   1   1   1   1
```

فإن العناصر الصفرية في المتجه المنطقي r تدل على $A \leq 5$ والعناصر التي تحتوي على الرقم واحد تدل على A>5 :

```
» r=(A==B)
r =
    0   0   0   0   0   0   0   1   0   0   0
```

في هذه المقارنة نبحث عن العناصر من A التي تساوي عناصر من B . كما يستخدم ~ لنفي الجملة :

```
» R=~(A>5)
R =
    1   1   1   1   1   0   0   0   0   0   0
» R=(A>5)&(A<8)
R =
      0   0   0   0   0   1   1   0   0   0
```

في هذه المقارنة ينتج العدد واحد عندما تكون A أكبر من 5 وأصغر من 8.

إضافةً لما سبق من عرض لأدوات المنطق الرياضي يوفر MATLAB للمستخدم العديد من الدوال المنطقية ، مثل الدوال في الجدول رقم (١.٥) :

الجدول رقم (١.٥). أدوات المنطق الرياضي.

دالة	xor(x,y)
تنتج صائب(1) إذا كان أي عنصر في x غير صفري	any(x)
تنتج صائب(1) إذا كان كل عنصر في x غير صفري	all(x)
صائب إذا كانت العبارة الحجية فارغة	isempty(x)
صائب إذا كان A و B متساويين	isequal(A,B)
صائب إذا كانت x مالا نهاية	isinf(x)
يبحث عن شرط معين	find

نمثل لطرق استخدام بعض هذه الدوال في التالي :

```
>> b=[0 inf 0 0 0  9];
>> find(isinf(b))
ans =
    2
```

الإجابة 2 تعطي رقم العنصر الذي يحقق الشرط :

```
>> a
a =
   0.5000    1.0000    1.6000    1.2000    0.8000    2.1000
>> find(isempty(a))
ans =
    [ ]
>> find(isequal(a,b))
ans =
    [ ]
```

نلاحظ إجابة MATLAB بـ [ ] لدالة البحث *find* ، مما يدل على أنه لا يوجد الشرط الذي نبحث عنه.

```
>> all(b)
ans =
   0
>> any(b)
ans =
   1
```

الإجابة 0 تدل على أنه ليس كل عناصر b صفرية ، أما الإجابة 1 تدل على وجود عناصر غير صفرية في b .

(١,١١) البرمجة في MATLAB

هناك تشابه كبير بين البرمجة في MATLAB ولغات البرمجة المعروفة ذات المستوى الرفيع. وقواعد MATLAB للبرمجة قريبة جداً من لغة FORTRAN و PASCAL مع بعض الإضافات من لغة C . يختلف MATLAB عن اللغات الأخرى بكونه بيئة interactive وجميع البرامج يتم ترجمتها في MATLAB جملة جملة interpreting بدلاً من ترجمة البرنامج compiling كما يجري في اللغات الأخرى. لأن MATLAB يتعامل مع المتغيرات على أنها مصفوفات ، فهو يوفر برامج عديدة متطورة و جاهزة للمستخدم في مجال المصفوفات مثل حلّ نظام معادلات خطية أو إيجاد القيم الذاتية لمصفوفة.

الأوامر for ، if و while تعدّ أساسيات البرمجة على MATLAB. نقوم بالبرمجة على MATLAB باستخدام الحلقات loops وهي مجموعة من الأوامر التي تنفّذ بشكل تكراري حتى يتحقق شرط معين مثل for loop ، while loop .

١ – **for :** يتم تنفيذ تكرار العبارة داخل الحلقة بعددٍ محدد من المرات يحدّد

بوساطة عدّاد الحلقة counter variable الذي يبين متغير العـدّاد، والقيمـة البدائيـة والقيمة النهائية للعداد.

الشكل العام :

```
for counter variable=initial value:final value
action
end
```

يمكننّا الأمر for من إنشاء المصفوفة c ذات الشكل الخاص بالبرنامج التالي :

```
» for i=1:5
for j=1:5
if i==j
c(i,j)=i*(5-i+1);
elseif j>i
c(i,j)=c(i,j-1)-i;
else j<i
c(i,j)=c(j,i);end
end
end
end
>> c
    5    4    3    2    1
    4    8    6    4    2
    3    6    9    6    3
    2    4    6    8    4
    1    2    3    4    5
```

لحساب مضروب الأعداد من ١الى ١٠ نكتب برنامجاً باستخدام *for* :

```
>> n=10;
>> fact=1;
>> for k=1:n
fact=k*fact;
disp([k fact])
end
    1      1
    2      2
    3     6
    4     24
    5     120
    6     720
```

```
    7     5040
    8     40320
    9     362880
    10    3628800
```

ولتوفير الجهد على المستخدم توجد دالة جاهزة في MATLAB تُدعى *factorial* لحساب مضروب أي عدد ، فمثلاً :

```
>> factorial(10)
ans =
    3628800
```

مثال رقم (١,٦)

استخدم حلقة for لحساب نهاية المتسلسلة $x_n = ax_{n-1} /n$ حيث a=10 (سنرى في الجزء القادم دالة جاهزة في MATLAB تقوم بحساب النهايات بأمر واحد). بما أن MATLAB لا يتعامل مع المتجهات غير المنتهية فإننا نستخدم قيمة كبيرة لعدد الحدود n لتظهر الصورة العامة للنهاية ، وهنا افترضنا أن n=10 حيث نرى نهاية العبارة تتقارب إلى 2.7557×10^3 :

```
>> a=10;
>> x=1;
>> k=10;

>> for n=1:k
x=a*x/n;
disp([n x])
end
    1        10
    2        50
    3        166.6667
    4        416.6667
    5        833.3333
    6        1.3889e+003
    7        1.9841e+003
    8        2.4802e+003
    9        2.7557 e+003
    10       2.7557e+003
```

٢- **While** : تستخدم لتنفيذ عبارات برمجية طالما أن الشرطَ محققُ.

الشكل العام :

```
while condition
action
increment action
end
```

في المثال نلاحظ استخدام أدوات الربط للمنطق الرياضي التي تسهل كتابة البرنامج :

```
while sum(x)~=max(y)
x=x.^2;
y=y+x;
end
```

٣- **if** : تقوم بفحص شرط منطقي وبعد ذلك يتم الانتقال إلى تنفيذ تعليمات معينة إذا تحقق الشرط. ونستطيع استخدام أكثر من عبارة if داخل بعضها على شكل nested loops .

الشكل العام :

```
if condition
statements
end
---------------
if condition
statements
else
statements
end
-------------
if  condition
<statements>
elseif  condition2
statements
else
statements
end
```

مثال رقم (١,٧)

لو أردنـــا حســــاب جــذور معـــــادلة من الدرجة الثــــانية ax^2+bx+c باستـخدام المميـز $\frac{-b \pm \sqrt{b^2-4ac}}{2a}$ للمعادلـة ($-10x+12+2x^2=0$) فإننـا نـدخل الخطوات التالية :

```
>> a=2;
>> b=-10;
>> c=12;
>> d=b^2-4*a*c;
>> if a~=0
if d<0
disp('complex roots')
else
x1=(-b+sqrt(d))/(2*a)
x2=(-b-sqrt(d))/(2*a)
end
end
```

نحصل على الجذور

```
x1 =
   3
x2 =
   2
```

(١,١٢) حسابات رمزية Symbolic Algebra

كما عرضنا في الأجزاء السابقة لـدى MATLAB القـدرة الفعّالة على التعامـل مع الحسابات العددية ، فإننا في هـذا الجـزء نعـرض قـدرة MATLAB علـى التعامـل مـع الحسابات الرمزية symbolic manipulations أيضاً . وهي التي تحتوي على رموز لـيس لها قيمٌ عددية معرفة مسبقاً ويعطي MATLAB نتائج بدلالـة هـذه الرموز. وهـذه الميـزة أضيفت في الإصدارات الجديدة من البرنامج ، ويتمكن MATLAB من إجراء ذلك عـن طريق المحرك MAPLE .

للـدخول في بيئـة الحـسابات بـالرموز يجـب طباعـة الأمـر *syms* وإدراج المتغيرات التي يراد استخدامها كرموز(يعتبر MATLAB الرموز ذات قيم صحيحة ، أما إذا كان

الرمز ذا قيمة حقيقية فيمكن إضافة كلمة real). فإذا أردنا حل معادلة من الدرجة الثالثة مثل $x^3-2x^2-x+2=0$ في x نستخدم الأمر *solve* :

```
>> syms x
>> solve(x^3-2*x^2-x+2)
```

والناتج هو الجذور الثلاثة :

```
ans =
 -1
 1
 2
```

أو إذا أردنا تحليل المعادلة نستخدم الأمر *factor* :

```
>> factor(x^3-2*x^2-x+2)
 ans =
 (x-1)*(x-2)*(x+1)
```

ويمكن إيجاد المشتقتين الأولى والثانية بالأمرين *diff(f(x))* و *diff(f(x),2)* :

```
>> diff(x^3-2*x^2-x+2)
ans =
3*x^2-4*x-1
>> diff(x^3-2*x^2-x+2,2)
 ans =
 6*x-4
```

أما حساب التكامل غير المحدود فيتم بالأمر *int(f(x))* :

```
>> int(x^3-2*x^2-x+2)
 ans =
 1/4*x^4-2/3*x^3-1/2*x^2+2*x
```

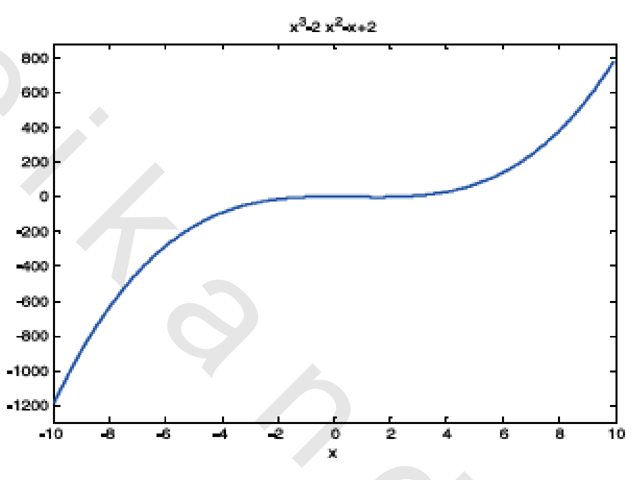
والتكامل المحدود من 0 إلى 3 لنفس الدالة هو :

```
>> int(x^3-2*x^2-x+2,0,3)
 ans =
 15/4
>>double(ans)
ans =
3.7500
```

لحساب القيمة العشرية للنتيجة السابقة

نستخدم الأمر **double (ans)**

كما يوجد أمرتنفيذي رمزي لرسم الدوال *ezplot(f,[a,b])*، حيث إن [a,b] هي الفترة المراد رسم الدالة عليها، فلنرسم دالتنا في الفترة [-10,10] لنصل للشكل رقم (١,١٧):



الشكل رقم (١,١٧). مثال لرسم بالأمر ezplot.

يقدم MATLAB دالة *dsolve* التي تعطي حلولاً لمعادلات تفاضلية. المعادلة ذات القيمة الابتدائية عند 1.5- :

$$y' = y - x, \quad \text{حيث إن} \quad y(-1.5) = -1$$

يمكن حساب الحل بتحديد المعادلة و القيمة الابتدائية و المتغير:

```
>> syms x y
>> h=dsolve('Dy=y-x,y(-1.5)=-1','x')
 h =
 x+1-1/2*exp(x)/exp(-3/2)
```

وإذا أردنا الحل العام لمعادلة تفاضلية من الدرجة الأولى $y' = x^2 + y$:

```
>> g=dsolve('Dy=x^2+y','x')
g =
-x^2-2*x-2+exp(x)*C1
```

أما لإيجاد نهاية العبارة الرياضية:

$$\lim_{x \to \infty} \frac{2x-1}{\sqrt{3x^2+x-1}}$$

فنكتب أمر *limit* مع تحديد الدالة ، والمتغير، والنهاية ، لنحصل على النتيجة:

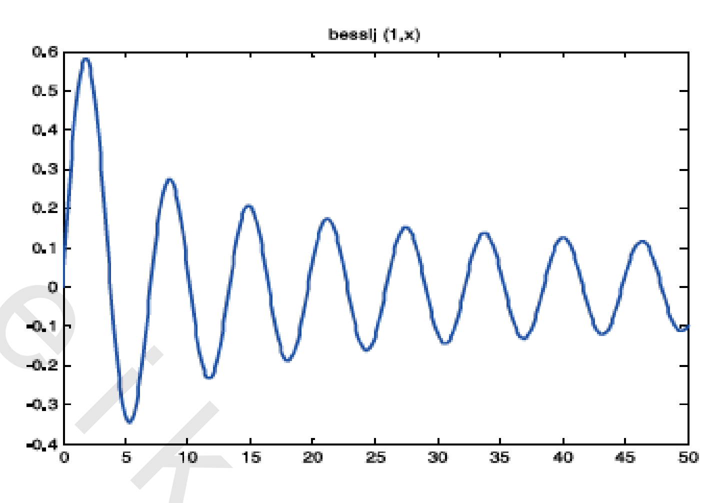
```
>> syms x, limit((2*x-1)/sqrt(3*x^2+x-1),x,inf)
ans =
2/3*3^(1/2)
```

أو النهاية من اليمين واليسار:

```
f =
(x-3)/abs(x-3)
>> a =limit (f,x,3,'left')
a =
-1
>> a =limit (f,x,3,'right')
a =
1
```

ويمكن تعريف دوال كثيرة جاهزة مثل laplace و الدالة العكسية ilaplace ، besselj أو gamma ورسمها بالشكل رقم (١,١٨) كذلك:

```
laplace(a)
ans =
1/s^2
>> laplace(t^2)
ans =
2/s^3
>> ilaplace(1/s^3)
ans =
1/2*t^2
besselj(1,[0:0.1:50])
plot([0:.1:50],besselj(1,[0:.1:50]))
```



الشكل رقم (١,١٨). مثال لرسم besselj .

وسيتم عرض كل هذه الأوامر الرمزية بتفصيل أكثر في الفصول القادمة حسب استخداماتها المختلفة. بصورة عامة في بيئة الحسابات الرمزية لاحظنا كيف يسّهل MATLAB إجراء الحسابات خاصة للطلاب في حلول التمارين المعقدة، والتي تستهلك الكثير من وقت الطالب لو أراد حلّها باليد، وبذلك يتمكن الطالب بوساطة MATLAB من الاطلاع على حلول لكمية أكبر من التمارين لترسيخ أي مفهوم رياضي.

(١,١٣) تمارين

١– اكتب العبارات التالية في أبسط صورة:

أ) $3.5-6/17(3^{2\pi})$

ب) $Sin(1.5)/5e^2$

جـ) 2>3&1

د) ~[1 2 0]*3

٢- المتجهان [3.2 1.25] و [1.9 5.5] يمثلان ضلعين في مثلث ، والضلع الثالث هو حاصل جمعهما ، احسب محيط المثلث.

٣- إذا كان A =[1 2 3 4 5] ، B =[4 5 6 7 8] ، C=[1 -1 2 -1 1] احسب A.*(B.*C) و قارن مع (A.*B).*C

٤- ارسم الدوال التالية :

أ) y=2sin3x+3cos2x $0 \le x \le 2\pi$

ب) $t=2s/(1+s^2)$ $-3 \le s \le 3$

جـ) y=3ln(1+x) $-1/2 \le x \le 2$

٥- اكتب m-file لرسم الدالة التالية :

$$f(x)=\begin{cases} x-3-\cos\frac{\pi}{4}x & 2 < x \le 4 \\ 1-\frac{x}{2}-\tan\frac{\pi}{8}x & 0 \le x \le 2 \end{cases}$$

٦- استخدم دالة mesh لرسم $z=2xy/(x^2+y^2)$ على x=1:0.1:3 y=1:0.1:3 .

ثم قارن بـ surf ، و contour .

٧- اكتب برنامجاً لحساب القيمة العظمى والقيمة الصغرى للدالة y = 2sin(2x)-3cos(x/2) على [0, 2 π] واستخدم الدوال min ، max والرسم للتأكد من النتائج.

٨- اكتب برنامجاً لاستخدام المميز لحساب جذري معادلة من الدرجة الثانية. طبق البرنامج على $2x^2-12x+18=0$ واستخدم fprintf لطباعة النتائج.

٩- اكتب برنامجاً لإنشاء مصفوفة A 3×3 قطرية بالقطر [4 2 1] ثم أجرِ العمليات التالية :

أ) بدل العمودين الثاني والثالث.

ب) قم بإضافة عمود رابع صفري وسمِ المصفوفة B .

جـ) احسب المحدد، الرتبة ، المنقول وأوجد المعكوس لـ A .

د) احسب A*B وقارن بـ A.*B

هـ) هل المصفوفة A متناظرة ؟

و) تأكد من $(AB)^t = B^tA^t$

١٠ - حلّ المعادلة $20x^2-31x-12 = 0$ باستخدام *syms* .

١١- بسّط العبارة $(x^{12}-y^{12})/(x-y)$ للحصول على كثيرة حدود.

١٢- استخدم *ezplot* لرسم الدالة $y=\exp(-x^{50})$ باختيار المجال المناسب.

١٣- احسب القيم الذاتية للمصفوفة :

$$A = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix}$$

١٤- أثبت باستخدام *inv* أن المصفوفة غير شاذة :

$$m = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & -1 \\ 3 & 1 & 1 \end{bmatrix}$$

١٥- ما هي قيم α ليصبح للمصفوفة التالية معكوس؟

$$A = \begin{bmatrix} 1 & -1 & a \\ 2 & 2 & 1 \\ 0 & a & -3/2 \end{bmatrix}$$

الفصل الثاني

حلول نظام المعادلات الخطية على MATLAB

(٢,١) نظام المعادلات الخطية Systems of Linear Equations

من أهم أسباب تطور علم الرياضيات هو البحث عن حلول لمسائل تطبيقية، وعندما نمثّل نظاماً تطبيقياً رياضياً فإننا أحيانا نلجأ لكتابته على شكل نظام معادلات خطية، وفي هذا الفصل سنعرض كيف يتم إيجاد حلول مباشرة لهذه الأنظمة على MATLAB، كما سنتطرق للطرق التكرارية في نهاية الفصل. يتم تمثيل المعادلات الخطية بمصفوفات ومتجهات وبذلك يكون MATLAB من أنسب البرامج لدراسة أنظمة المعادلات الخطية.

بصفة عامة، لإنشاء نظامٍ من n معادلات خطية في n متغيرات $x_1, x_2, \ldots x_n$ نكتب:

$$\begin{matrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{matrix}$$

حيث يُمثل النظام كمصفوفة بـ Ax=b . وترمز A للمصفوفة المربعة بحجم nxn المكونة من المعاملات، و b متجه الطرف الأيمن بقياس n ونبحث عن الحل المتعلق بمتجه المجهول $x_1, x_2, \ldots x_n$.

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

إذا فرضنا أن مصفوفة المعاملات A هي مصفوفة غير شاذة، أي يوجد لها معكوس، فبذلك يكون للنظام حل وحيد unique solution ، ويمثل بالمعادلة $x=A^{-1}b$. أما إذا كانت A غير مربعة و عدد المعادلات أكثر من عدد المتغيرات فإن النظام يُعدّ over-determined ولا يوجد حل له. وإذا كان عدد المعادلات أقل من عدد المتغيرات في النظام فإنه يُعدّ under-determined و يوجد عدد غير منتهٍ من الحلول.

(٢,٢) حل نظام المعادلات الخطية Ax=b باستخدام \ على MATLAB

يتم حل نظام معادلات خطية على **MATLAB** باستخدام أداة القسمة باليسار \ وهي تختلف عن القسمة باليمين / بالنسبة للمتجهات والمصفوفات. فعند ادخال A\B فهذا يكافئ inv(A)*B ، أما B/A فهو يكافئ B*inv(A) .

مثال رقم (٢,١)

نفرض أن لدينا النظام :

$$\begin{aligned} 3x_1+2x_2-x_3&=10 \\ -x_1+3x_2+2x_3&=5 \\ x_1-x_2-x_3&=-1 \end{aligned}$$

ندخل في **MATLAB** مصفوفة المعاملات A ومتجه الطرف الأيمن b :

```
>> A
A =
    3    2   -1
   -1    3    2
    1   -1   -1
>> b=[10 5 -1]'
b =
   10
    5
   -1
>> A\b
ans =
  -2.0000
   5.0000
  -6.0000
```

بتطبيق عملية القسمة من اليسار نحصل على الحل $x_1= -2, x_2 = 5, x_3= -6$.

يمكن أيضاً استخدام المعكوس لإيجاد الحل بحساب $A^{-1}b$ ، ولكن الحل بالأداة \ أدق وأقل استهلاكاً للعمليات الحسابية .

```
>> inv(A)*b
ans =
  -2.0000
   5.0000
  -6.0000
```

عند استعمال عملية القسمة باليسار \ لحل نظام Ax=b فإن MATLAB يختار الطريقة الأنسب والأقل تكلفة حسب نوع المصفوفة A :

- إذا كانت A مصفوفة مثلثية (علوية أو سفلية) فإن MATLAB يستخدم التعويض التراجعي أو الأمامي فقط Backward or Forward substitution .
- إذا كانت A مصفوفة مربعة فإن MATLAB يستخدم الحذف الجاوسي Gaussian elimination .
- إذا كانت A مصفوفة غير مربعة فإن MATLAB يستخدم التحليل QR Factorization .

- إذا كانت A مصفوفة مربعة و sparse فإن MATLAB يستخدم التحليل Cholesky Factorization .

مثال رقم (٢.٢)

إذا كان عدد المعادلات أكثر من عدد المتغيرات في النظام over-determined system مثل :

$$x_1 + x_2 = 2$$
$$2.05x_1 - x_2 = 1$$
$$3.06x_1 + x_2 = 3.5$$
$$-x_1 + 2x_2 = 0.92$$
$$4x_1 + x_2 = 3$$

فإن MATLAB يقدّم الحل بطريقة التقريب بأصغر مربعات least squares approximation. ندخل مصفوفة المعاملات ومتجه اليمين ونستخدم القسمة باليسار :

```
>> c=[1 1 ;2.05 -1;3.06 1;-1 2;4 1]
c =
   1.0000    1.0000
   2.0500   -1.0000
   3.0600    1.0000
  -1.0000    2.0000
   4.0000    1.0000
>> d=[2;1;3.5;.92;3]
d =
   2.0000
   1.0000
   3.5000
   0.9200
   3.0000
>> c\d
ans =
   0.7159
   0.8087
```

ولأن النظام غير متّسق inconsistent system فإن الحل سيكون تقريبياً لبعض المعادلات وليس لكلها. في المقابل إذا كان عدداً المتغيرات أكثر من عدد المعادلات under-determined system فالناتج سيكون عدد غير منتهي من الحلول.

مثال رقم (٢,٣)

أوجد حل النظام التالي :

$$x_1 + 2x_2 + 3x_3 = 1$$
$$-4x_1 + 2x_2 + 5x_3 = 3$$

```
>> a=[1 2 3;-4 2 5]
a =
   1    2    3
  -4    2    5
>> b=[1;3];
>> a\b
ans =
 -0.2353
       0
  0.4118
```

يحسب MATLAB الحل بأداة \ ، و نلاحظ أن MATLAB عيّن القيمة صفر اختيارياً للمتغير x_2 ، و لم يتم تحذير المستخدم على أن هذا الحل هو واحد فقط من بين عدد غير منتهٍ من الحلول .

(٢,٢,١) الصيغة الدرجية الصفية المختزلة (RREF)

توجد لدى MATLAB دالة *rref* ، وهي تحول المصفوفة إلى الصيغة الدرجية الصفية المختزلة Reduced Row Echelon Form (RREF) . وهذه الصيغة تتحقق بالمواصفات التالية :

١- في كل صف غير صفري يجب أن يكون أول عنصر غير صفري فيه يساوي ١.

٢- الصفوف الصفرية (إن وجدت) يجب أن تكون في أسفل المصفوفة.

٣- إذا وجد صفان غير صفريين فإن العنصر المتقدم ١ في الصف الأعلى يجب أن يكون على يسار العنصر المتقدم ١ في الصف الأسفل، ويكون باقي العمود (الذي يحتوي على ١) أصفاراً.

لنظام المعادلات Ax=b ننشئ المصفوفة الموسّعة [A b] augmented matrix بضم المصفوفة A مع المتجه b، وإذا تم تحويل هذه المصفوفة إلى شكلها RREF فيمكن استنتاج التالي:

- إذا صدرت [A b] من نظام غير متسق inconsistent system فإن في مصفوفة RREFسيكون هناك صفٌ على شكل [0...01] .
- إذا صدرت [A b] من نظام متسق consistent system بعدد غير منتهٍ من الحلول فإنه في مصفوفة المعاملات في RREF سيكون عدد الأعمدة أكثر من عدد الصفوف غير الصفرية، أو أن هناك حلاً وحيداً للنظام، وسيظهر في آخر عمود في RREF .
- إذا ظهر صف صفري في مصفوفة فهذا يدل على أن النظام الأصلي يحتوي على معادلة مكررة .

نستنتج من ذلك أن في نظام متسق Ax=b وفي حال كون A مصفوفة مربعة مع وجود حل وحيد، فإن الشكل RREF للمصفوفة A هو مصفوفة الوحدة. المثال (٢،٤) يوضح ذلك :

مثال رقم (٢,٤)

إذا كان لدينا نظام Ax=b حيث إن A و b:

```
A =
   8   1   6
   3   5   7
   4   9   2
>> b=ones(3,1)
b =
   1
   1
   1
```

استخدام *rref* على المصفوفة [A b] يعطي مصفوفة الوحدة، و عمود الحل هو العمود الذي يمكن فصله عن طريق *x(:,4)*:

```
>> [x, pivot]=rref([A b])
x =
   1.0000        0        0   0.0667
        0   1.0000        0   0.0667
        0        0   1.0000   0.0667
pivot =
   1   2   3
>> x=x(:,4)
x =
   0.0667
   0.0667
   0.0667
```

أما المتجه pivot الناتج من *rref* فيخزّن أماكن عمود المحورة pivot columns indices ويمكن استخدامه لحساب رتبة A، وللتأكد نحسب رتبة المصفوفةA بالأمر *rank*:

```
>> length(pivot)
ans =
   3
>> rank(A)
ans =
   3
```

استخدام آخر لدالة *rref* هو إيجاد معكوس المصفوفةA وذلك بتطبيق دالة *rref* علـى المـصفوفة الموسـعة لـ A ومـصفوفة الوحـدة. فيظهـر معكـوسA في الأعمـدة الأخيرة:

```
H=rref([A eye(size(A))])
H =
    1    0    0   -1    3    7
    0    1    0    1   -2   -5
    0    0    1   -2    5   11

>>B=H(:,4:6)
B =
   -1    3    7
    1   -2   -5
   -2    5   11
```

للتأكد من المعكوس B نحسب A*B=B*A=I :

```
>> B*A
ans =
    1    0    0
    0    1    0
    0    0    1

>> A*B
ans =
    1    0    0
    0    1    0
    0    0    1
```

أو عن طريق دالة *inv(A)* التي تطابق المصفوفة *B*:

```
>> inv(A)

ans =
  -1.0000    3.0000    7.0000
   1.0000   -2.0000   -5.0000
  -2.0000    5.0000   11.0000
```

كما يوجد الأمر *rrefmovie* الذي يُمكن المستخدم من رؤية المصفوفات الناتجة في كل خطوة من عملية الاختزال، و ذلك بالضغط على أي حرف من لوحة المفاتيح حتى يصل للصيغة النهائية، مثل :

```
Original matrix

A =
     8        1        6
     3        5        7
     4        9        2
Press any key to continue. . .
 pivot = A(1,1)

A =
     1        1/8       3/4
     3        5         7
     4        9         2
Press any key to continue. . .
```

(٢,٢,٢) دالة Solve

يوجد في البيئة الرمزية symbolic إمكانية حل نظام من المعادلات من ذوات الحلول الفعلية exact solutions ، وذلك بالأمر solve، الذي يستخدم على المعادلة لينتج الحلول.

مثال رقم (٢,٥)

لحل النظام في البيئة الرمزية :

$$x + 2y = 8$$
$$3x + 4y = 18$$

```
>> syms x y
>> [x0,y0]=solve(x+2*y-8,3*x+4*y-18)
x0 =
2
y0 =
 3
```

(٢,٣) حل نظام المعادلات الخطية بالحذف الجاوسي Gaussian Elimination

الحذف الجاوسي طريقة عملية لحل نظام معادلات، خاصةً الأنظمة ذات المعاملات الصفرية القليلة. وتعتمد الطريقة على عمليات التبسيط الثلاث الأساسية على صفوف النظام (المعادلات):

١- التبديل بين الصفوف .

٢- ضرب الصف بعدد ثابت غير صفري.

٣- التعويض عن صف بحاصل جمع الصف ذاته وصف آخر مضروب بعدد ثابت.

باختصار، طريقة الحذف الجاوسي تُحول المصفوفة الموسعة للنظام إلى مصفوفة مثلثية علوية، ومن ثم نستخدم التعويض التراجعي لحساب قيم المتغيرات.

مثال رقم (٢,٦)

نفرض أن لدينا النظام:

$$
\begin{aligned}
3x_1 + 2x_2 - x_3 &= 10 \\
-x_1 + 3x_2 + 2x_3 &= 5 \\
x_1 - x_2 - x_3 &= -1
\end{aligned}
$$

ننشئ المصفوفة الموسعة:

$$
[A \ \ b] = \begin{bmatrix} 3 & 2 & -1 & 10 \\ -1 & 3 & 2 & 5 \\ 1 & -1 & -1 & -1 \end{bmatrix}
$$

وهنا نستطيع استخدام *rref* على هذه المصفوفة ، وهذا موازٍ لعمل الحذف الجاوسي مع التعويض التراجعي وسيظهر متجه الحل في العمود الرابع :

```
>> A=[3 2 -1 10;-1 3 2 5;1 -1 -1 -1];
>> G=rref(A)
G =
   1    0    0   -2
   0    1    0    5
   0    0    1   -6

>> x=G(:,4)
x =
  -2
   5
  -6
```

بطريقة أخرى يمكن كتابة m-file لبرنامج Gaussian [7] ويتم حفظه تحت اسم Gaussian.m (خوارزمية ٢،١) مع مراعاة أن يبدأ الملف بكلمة function :

```
function x=Gaussian(B)
[n,t]=size(B);G=B;
for i=1:n-1
  for j=i:n-1
     m=G(j+1,i)/G(i,i);
   for k=1:t
       G(j+1,k)=G(j+1,k)-m*G(i,k);
   end
  end
end
j=n;x(j,1)=G(j,t)/G(j,j);
  for j=n-1:-1:1
   w=0;
   for k=n:-1:j+1
    w=w+G(j,k)*x(k,1);
   end
   x(j,1)=(G(j,t)-w)/G(j,j);
 end
disp(G)
```

خوارزمية (٢،١).

ويتم استخدام البرنامج Gaussian المعطى في الخوارزمية (٢،١) على المصفوفة الموسعة A للمثال السابق بكتابة الأمر *Gaussian(A)* لنحصل على المصفوفة الناتجة من إجراء خطوات الحذف الجاوسي ثم متجه الحل.

```
>> Gaussian(A)

  3.0000   2.0000  -1.0000  10.0000
       0   3.6667   1.6667   8.3333
       0        0   0.0909  -0.5455

 -2.0000
  5.0000
 -6.0000
```

عند وجود العدد صفر على قطر المصفوفة يمكن تبديل ترتيب المعادلات لتفادي ذلك قبل إجراء الحذف الجاوسي. وهناك طرق للمحورة pivoting strategies يتم إجراؤها مع الحذف الجاوسي ليس فقط لمنع ظهور الأصفار على القطر ولكن لوضع أكبر عدد في كل صف على القطر وذلك لتحاشي القسمة على عدد صغير، مما يؤدي إلى تراكم أخطاء التدوير، وتدعى هذه الطريقة بمحورة جزئية partial pivoting . وهناك أيضا محورة كاملة total pivoting التي يتم فيها المحورة على كل المصفوفة أي يمكن أن يتم التبادل بين الأعمدة أيضا حسب الحاجة. يمكن للقارىء كتابةm-files على MATLAB لبرامج الحذف الجاوسي مع محورة جزئية أو كاملة ومن ثم تطبيقها.

(٢،٤) حل نظام المعادلات الخطية بالتحليل Factorization

توجد طريقة أخرى مباشرة لحل النظام Ax=b وهي إعادة كتابة المصفوفةA على شكل جداء مصفوفتين، سنعرض في هذا الجزء بعض الجداءات المختلفة.

(٢,٤,١) التحليل A=LU

بحيث تكون L مصفوفة مثلثية سفلية وU مصفوفة مثلثية علوية، وكلاهما بنفس حجم A .

$$A = LU = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{12} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

تُعد طريقة التحليلLU طريقةً مباشرةً لحل نظام معادلات خطية، وتفيد في حال وجود أكثر من متجه b في الطرف الأيمن أو إذا كان b غير معلوم، لأن إيجاد L و U لا يعتمد على الطرف الأيمن b. الطريقة تعتمد على تحويل النظام Ax=b إلى LUx=b ، ثم نفرض أن y=Ux لتنتج Ly=b ، وهنا نستخدم التعويض الأمامي لأن L مصفوفة مثلثية سفلية. ولإيجاد متجه الحل x للنظام نستخدم التعويض التراجعي لحل Ux=y لأن U مصفوفة مثلثية علوية .

كما أن هناك عدة أشكال للتحليل على حسب اختيار قيم القطر للمصفوفتين L و U:

- إذا وضعنا قيم قطر L العدد 1 فإن التحليل يدعى طريقة دووليتل Doolittle's method .
- إذا وضعنا قيم قطر U العدد 1 فإن التحليل يدعى طريقة كراوت Crout's method .
- إذا كان النظام متناظراً symmetric و positive definite معرّفة إيجابيا أي $x^t A x > 0$ لكل متجه غير صفري x ، فإن التحليل $A=L L^t$ يدعى طريقة شلوسكي Cholesky method .

(٢,٤,٢) حل نظام معادلات خطية بتحليل A=LU

يقوم MATLAB بإيجاد التحليل A=LU بالأمر lu(A) ، فمثلاً تحليل المصفوفة التالية :

```
>>A =
   3    2   -1
  -1    3    2
   1   -1   -1

>> [L,U]=lu(A)

L =
   1.0000        0         0
  -0.3333   1.0000         0
   0.3333  -0.4545    1.0000

U =
   3.0000   2.0000   -1.0000
        0   3.6667    1.6667
        0        0    0.0909
```

ينتج lu المصفوفة السفلية

والمصفوفة العلوية

إذا حددنا المتجه اليمين b في النظام Ax=b :

```
>> b=[10 5 -1];
```

و نبدأ بالخطوة الأولى : حل النظام الأول Ly=b باستخدام الأداة \ لإيجاد المتجه y :

```
>> y=L\b'
y =
  10.0000
   8.3333
  -0.5455
```

الخطوة الثانية : إيجاد متجه الحل x بحل النظام Ux = y :

```
>> x=U\y
x =
-2.0000
5.0000
-6.0000
```

نلاحظ أن MATLAB يستخدم تحليل دووليتل Doolittle's ، وإذا أردنا تحليل كراوت Crout's فيجب كتابته في m-file .

(٢,٤,٣)حل نظام معادلات خطية تحليل شلوسكي $B = L^tL$

يوجـد في MATLAB تحليـل شلوسـكي cholesky بالدالـة الجـاهزة *chol(A)* . ويتوجـب على المصفوفة أن تكون متناظرة symmetric ومعرّفة إيجابياً positive definite مثل :

```
>> B
B =
  2.0000          0 - 1.0000i     0
 0 + 1.0000i      2.0000          0
 0                0               3.0000
```

إذا طلبنا التحليل الشلوسكي للمصفوفة نحصل على المصفوفة L ، وللتأكد من التحليل نحسب $B = L^tL$:

```
>> L=chol(B)
L =
  1.4142          0 - 0.7071i     0
       0          1.2247          0
       0          0               1.7321

>> L'*L
ans =

   2.0000          0 - 1.0000i     0
   0 + 1.0000i     2.0000          0
   0               0               3.0000
```

مثال رقم (٢،٦)

إذاعرّفنا المتجه الأيمن b = (1, 2, 3) نستطيع إيجاد حل النظام B x =b، بحل $L^t y = b$ ثم L x = y باستخدام \:

```
>> y=L'\b'
y =
  0.7071
  1.6330 - 0.4082i
  1.7321

>> L\y
ans =
  0.6667 + 0.6667i
  1.3333 - 0.3333i
  1.0000
```

(٢،٤،٤) تحليل A = QR

يوفّر MATLAB دالة *qr* لتحليل A = Q R بحيث تكون R مصفوفة مثلثية علوية و Q مصفوفة متعامدة، (تكون Q متعامدة إذا $Q^t = Q^{-1}$)، وليس من الضروري أن تكون المصفوفة A مربعة. فيصبح الحل للنظام A x = b هو الحل للنظام $R x = Q^t b$.

```
>> A=[4 -2 7;6 2 -3; 3 4 5];
>> [Q,R]=qr(A)
Q =
  -0.5121   0.6852   0.5179
  -0.7682  -0.0958  -0.6330
  -0.3841  -0.7220   0.5754

R =
 -7.8102  -2.0486  -3.9691
       0  -4.4501   0.0295
       0        0   9.5522
```

(٢,٤,٥) تحليل القيمة الشاذة svd

يوفر MATLAB دالة svd التي تقدم تحليل القيمة الشاذة singular value decomposition، A=USV لمصفوفة A بحجم mxn و بحيث تكون مصفوفةU متعامدة بحجم mxm ، وV مصفوفة متعامدة بحجم nxn و مصفوفة S قطرية بحجم mxn ، كما أن القيم على قطر S تسمى القيم الشاذة وعددها يساوي رتبة المصفوفة . يُعد هذا النوع من التحليل الأكثر ضماناً ولكنه يحتاج إلى كمية حسابات أكثر من غيره. يتم استخدام svd غالباً في حل مسائل أصغر المربعات least squares problems والطرق المثلى .

لحساب تحليل svd للمصفوفة A :

```
>> A=[1 2 3;4 5 9;7 11 18;-2 3 1;7 1 9]
A =
    1    2    3
    4    5    9
    7   11   18
   -2    3    1
    7    1    9

>> [u,s,v]=svd(A)
u =
  -0.1364   0.0871   0.0284  -0.2001  -0.9659
  -0.4069   0.0334  -0.2544  -0.8465   0.2283
  -0.8161   0.2947  -0.1691   0.4656   0.0404
  -0.0511   0.5609   0.8018  -0.1632   0.1152
  -0.3836  -0.7680   0.5128   0.0000   0.0000

s =
  27.1420        0          0
        0   6.1825          0
        0        0     0.2968
        0        0          0
        0        0          0

v =
  -0.3706  -0.6816  -0.6309
  -0.4355   0.7275  -0.5301
  -0.8203  -0.0783   0.5665
```

وإذا أدخلنا *svd(A)* فنحصل على القيم القطرية فقط، أي القيم الشاذة مباشرة:

```
>> svd(A)
ans =
  27.1420
   6.1825
   0.2968
```

(٢,٥) طرق تكرارية Iterative Methods

في حالة وجود أنظمة خطية صغيرة فالطرق المباشرة السابقة تكون مناسبةً، ولكن للأنظمة الكبيرة و المحتوية على معاملات صفرية عديدة فالطرق التكرارية تكون أكثر عملية من ناحية استهلاك ذاكرة الحاسوب والدقة. الطرق التكرارية لحل النظام Ax=b تبدأ بقيمة تقريبية ابتدائية $x^{(0)}$ لحل النظام x ، وتولّد متتالية $\{x^{(k)}\}_{k=0}^{\infty}$ من المتجهات التي تتقارب من x . معظم الطرق التكرارية تحّول النظام إلى نظام مكافئ بالشكل x = Tx+c لمصفوفة مربعة T و متجه c . بعد اختيار المتجه الابتدائي $x^{(0)}$ نقوم بتوليد متتالية لمتجهات الحلول التقريبية من $x^{(k+1)} = Tx^{(k)} + c$ لكل k=0,1,2,... . ونتوقف عن التكرار إذا كان الخطأ صغيراً جداً بين المتجهات التقريبية المتتالية أي $\left\| x^{(k+1)} - x^{(k)} \right\| < \varepsilon$ لعدد صغير موجب ε . من بين الطرق التكرارية الأكثر شيوعاً طريقة جاكوبي التكرارية Jacobi iterative method و طريقة جاوس سيدال التكرارية Gauss-Seidel iterative method .

طريقة جاكوبي التكرارية تحل المعادلة رقم *i* في النظام للحصول على x_i :

$$x_i = \sum_{j=1, i\neq i}^{n} \left(-\frac{a_{ij}x_j}{a_{ii}}\right) + \frac{b_i}{a_{ii}} \qquad i = 1,2,3\ldots,n$$

وتولد $x_i^{(k)}$ باستخدام $x^{(k-1)}$ لكل $k \geq 1$ عن طريق :

$$x_i^{(k)} = \frac{\sum_{j=1, j\neq j}^{n}(-a_{ij}x_j^{(k-1)}) + b_i}{a_{ii}} \qquad i = 1,2,3\ldots,n$$

أما طريقة جاوس سيدال فتستخدم:

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1}(a_{ij}x_j^{(k)}) - \sum_{j=i+1}^{n}a_{ij}x_j^{(k-1)} + b_i}{a_{ii}} \qquad i = 1,2,3\ldots,n$$

والفرق بين طريقة جاكوبي التكرارية وطريقة جاوس سيدال التكرارية هو أن الأخيرة تستخدم القيم الجديدة لـ x_i كلما حُسبت. يمكن برمجة الطرق التكراريـــــة في m-file [7] واستخدامها كدالة للحصول على الحل التقريبي للنظام. الخوارزمية (٢،٢) تعطي برنامج GaussSeidel هي:

```
function x=GaussSeidel(B,x,tol)
[n,t]=size(B);
b=B(1:n,t); w=1;k=1;
d(1,1:n+1)=[0 x]; k=k+1;
while w>acc
for i=1:n
sum=0;
for j=1:n
if j<=i-1
    sum=sum+B(i,j)*d(k,j+1);
elseif j>=i+1
sum=sum+B(i,j)*d(k-1,j+1);
end ;end;
x(1,i)=(1/B(i,i))*(b(i,1)-sum);
d(k,1)=k-1;d(k,i+1)=x(1,i);
end
w=max(abs((d(k,2:n+1)-d(k-1,2:n+1))));
k=k+1;
if w>100 & k>10
   ('Gauss-Seidel method is Divergent')  end;end;x=d;
```

خوارزمية (٢،٢).

مثال رقم (٢,٧)

أوجد حل النظام بالطرق التكرارية :

$$5x_1 - 1x_2 + x_3 = 10$$
$$2x_1 + 8x_2 - x_3 = 11$$
$$-x_1 + x_2 + 4x_3 = 5$$

نُدخل المصفوفة الموسعة للنظام و المتجه الابتدائي و الدقة المطلوبة للحل :

```
>> B
B =
    5   -1    1   10
    2    8   -1   11
   -1    1    4    5
>> x
x =
    0    0    0
>> tol
tol =
  1.0000e-006
```

نستعمل برنامج جاكوبي لإيجاد حل النظام السابق ، بإدخال *Jaccobi(B,x,tol)* (الملحق) [7] بالمدخلات اللازمة ، و يعرضMATLAB النتائج في العمود الأول ، رقم خطوة التكرار ، والأعمدة الثلاثة التالية تعطي إحداثيات متجه الحل .

نلاحظ أن طريقة جاكوبي اســتغرقت ١٦ خطـــوة تكراريـــة لإيجاد الحـــل x= [1.923, 1.076, 1.461] بالدقة المطلوبة.

```
Jaccobi(B,x,tol)
ans =    k           x1         x2         x3
     0           0          0          0
    1.0000    2.0000    1.3750    1.2500
    2.0000    2.0250    1.0313    1.4063
    3.0000    1.9250    1.0445    1.4984
    4.0000    1.9092    1.0811    1.4701
```

5.0000	1.9222	1.0815	1.4570
6.0000	1.9249	1.0766	1.4602
7.0000	1.9233	1.0763	1.4621
8.0000	1.9228	1.0769	1.4617
9.0000	1.9230	1.0770	1.4615
10.0000	1.9231	1.0769	1.4615
11.0000	1.9231	1.0769	1.4615
12.0000	1.9231	1.0769	1.4615
13.0000	1.9231	1.0769	1.4615
14.0000	1.9231	1.0769	1.4615
15.0000	1.9231	1.0769	1.4615

للمقارنة ، نكتب الأمر *GaussSeidel* بالمدخلات اللازمة. ونلاحظ أن طريقة جاوس سيدال التكرارية احتاجت فقط ١١ خطوة تكرارية للوصول للحل بنفس الدقة :

```
>> GaussSeidel(B,x,tol)
```

ans =

k	x_1	x_2	x_3
0	0	0	0
1.0000	2.0000	0.8750	1.5313
2.0000	1.8688	1.0992	1.4424
3.0000	1.9314	1.0725	1.4647
4.0000	1.9215	1.0777	1.4610
5.0000	1.9233	1.0768	1.4616
6.0000	1.9230	1.0769	1.4615
7.0000	1.9231	1.0769	1.4615
8.0000	1.9231	1.0769	1.4615
9.0000	1.9231	1.0769	1.4615
10.0000	1.9231	1.0769	1.4615

وإذا كانت كل من الطريقتين جاكوبي و جاوس سيدال تتقارب من متجه الحل ، فإن جاوس سيدال هو الأسرع ، ولكن ليس دائماً لأن هناك أنظمة تكون فيها طريقة جاكوبي متقاربة ، ولكن جاوس سيدال متباعدة ، أو العكس. لنضمن التقارب لكل من الطريقتين ، يجب التحقق من شرط في النظام وهو أن تكون مصفوفة المعاملات للنظام $A=(a_{ij})$ مربعة بحجم nxn ومسيطرة قطرياً بدقة strictly diagonally dominant أي :

$$|a_{ii}| > \sum_{\substack{j=1 \\ i \neq j}}^{n} |a_{ij}| \quad \text{for } i = 1,2,\ldots,n$$

(٢,٦) مسائل القيم الذاتية Eigenvalue problem

تظهر مسائل القيم الذاتية في كثير من تطبيقات الجبر الخطي في فروع العلوم الطبيعية والهندسة، وصيغة هذا النوع من المسائل العامة تأخذ الشكل $Ax=\lambda x$ وهي معادلة جبرية للقيم الذاتية لـA ، حيث إن A مصفوفة مربعة بحجم nxn ونقول ان العدد λ قيمة ذاتية أومميزة Eigenvalue للمصفوفة A إذا وجد متجه x غير صفري يسمى متجهاً ذاتياً Eigenvector بحيث إن $Ax=\lambda x$ وهي . ويمكن إعادة كتابة مسألة القيمة الذاتية على شكل نظام المعادلات $(A-\lambda I)x=0$ وهي حيث إن I هي مصفوفة الوحدة بحجم nxn . ويكون هناك حلٌ للنظام إذا وإذا فقط كان $(A-\lambda I)$ نظاماً شاذاً أو $det(A-\lambda I)=0$. وهذه المعادلة هي كثيرة حدود بدرجة n في المتغير λ وتسمى المعادلة الذاتية characteristic equation لـ A. جذور المعادلة الذاتية هي القيم الذاتية $\lambda_1, \lambda_2, \ldots, \lambda_n$ ولكن المتجه الذاتي x المقابل لكل قيمة ذاتية λ_i ليس وحيداً.

في برنامج MATLAB نقوم بحساب القيم الذاتية بالأمر *eig* كما في المثال التالي.

مثال رقم (٢,٩)

إذا كان لدينا مصفوفة A :

```
>> A= [-6 0 0;11 -3 0;-3 6 7]
>> [X,D]=eig(A)
X =
      0        0          0.2348
      0        0.8575    -0.8608
   1.0000  -0.5145    0.4515
```

```
D =
    7    0    0
    0   -3    0
    0    0   -6

>> lambda=eig(A)
lambda =
    7
   -3
   -6
```

ينتج من الأمر *[X,D]=eig(A)* مصفوفتان X و D . الأعمدة في المصفوفة X تحتوي على المتجهات الذاتية المقابلة للقيم الذاتية التي تظهر على قطر المصفوفة D، ويمكن إيجاد القيم الذاتية مباشرة بالدالة *eig(A)* فقط، وتخزينها في متجه lambda . لإيجاد كثيرة الحدود المميزة نوجد أولاً المعاملات بدالة *poly* ثم بدالة *poly2sym* التي تحولها إلى معادلة في المتغير x .

```
>> coefChar=poly(A)
coefChar =
    1    2  -45  -126
>> CharEq=poly2sym(coefChar)
CharEq =
x^3+2*x^2-45*x-126
```

لقد عرضنا في هذا الباب أهم الطرق لحل أنظمة المعادلات الخطية، وقدرة **MATLAB** على معالجتها بصورة عملية و دقيقة. يستطيع القارئ تطوير البرامج التي عرضت لاستخدامها في تطبيقات أخرى، ومن ثم تسخيرها في إيجاد حلول لمسائل فيزيائية وهندسية مختلفة.

تمارين (٢،٧)

١- أوجد حل الأنظمة الخطية التالية باستعمال دالة " \ " وقارن باستخدام المعكوس:

أ)
$$5x_1+3x_2+x_3=3$$
$$2x_1+3x_2+x_3=-1$$
$$x_1+x_2+3x_3=2$$

جـ)
$$2x_1-x_2+x_3=-1$$
$$3x_1+3x_2+9x_3=0$$
$$3x_1+3x_2+5x_3=4$$

ب)
$$x_1+3x_2-x_3=4$$
$$2x_1+2x_2+x_3=9$$
$$5x_1-2x_2-x_3=-2$$

د)
$$2x_1=3$$
$$x_1+1.5x_2=4.5$$
$$-3x_2+0.5x_3=-6.6$$
$$2x_1-2x_2+x_3+x_4=0.8$$

٢- حل الأنظمة السابقة بالتحليل $A=LU$ إن أمكن .

٣- حل الأنظمة السابقة بالحذف الجاوسي.

٤- أوجد حل النظام التالي بالحذف الجاوسي ، مع المحورة الجزئية ، وقارن الحل بالمحورة الكاملة :

$Ax=b$ إذا $a_{ij}=1/(i+j-1)$ لكل $i,j=1,2,....,n$.

٥- استخدم طريقة جاوس سيدال تكرارية لإيجاد حل النظام :

$$2x_1+x_2+4x_3=16$$
$$4x_1+2x_2+x_3=11$$
$$-x_1+2x_2=3$$

مبتدئاً بالمتجه $x=[1,1,1]^t$.

٦- قارن حل النظام السابق بطريقة جاكوبي التكرارية.

٧- أوجد التكرارين الأولين من طريقة جاكوبي باستخدام $\mathbf{x}^{(0)}=\mathbf{0}$:

أ)
$$10x_1-x_2=1$$
$$-x_1+10x_2-2x_3=7$$
$$-2x_2+10x_3=6$$

ب)
$$3x_1-x_2+x_3=1$$
$$3x_1+6x_2+2x_3=0$$
$$3x_1+3x_2+7x_3=4$$

جـ)

$$\begin{aligned} 4x_1 + x_2 - x_3 + x_4 &= -2 \\ x_1 + 4x_2 - x_3 - x_4 &= -1 \\ -x_1 - x_2 + 5x_3 + x_4 &= 0 \\ x_1 - x_2 + x_3 + 3x_4 &= 1 \end{aligned}$$

د)

$$\begin{aligned} 10x_1 + 5x_2 &= 1 \\ 5x_1 + 10x_2 - 4x_3 &= 25 \\ -4x_2 + 8x_3 - x_4 &= -11 \\ -x_3 + 5x_4 &= -11 \end{aligned}$$

٨- أوجد التكرارين الأولين في طريقة جاوس سيدال باستخدام $\mathbf{x}^{(0)}=\mathbf{0}$ للأنظمة في التمرين رقم ٧ .



الفصل الثالث

حل المعادلات غير الخطية على MATLAB

في هذا الفصل يتم عرض إحدى المسائل الأساسية في الرياضيات، وهي إيجاد الحل العددي لمعادلة غير خطية. معظم المعادلات التي تظهر في التجارب العلمية أو في الطبيعة هي معادلات غير خطية، وقد تكون المعادلات في متغير واحد أو أكثر، وقد تكون معادلة واحدة أو نظاماً من المعادلات. الطرق العددية تقرّب حل المعادلات غير الخطية بطريقة تكرارية، ويكون التقارب في بعضها مضموناً، وفي بعضها الآخر مشروطاً.

إذا فرضنا أن $f(x)$ دالة متصلة، فإن العدد α بحيث إن $f(\alpha) = 0$ يسمى جذراً أو صفراً للمعادلة $f(x) = 0$. و يمكن أن يكون الجذر حقيقياً أو مركباً، كما يمكن أن يوجد أكثر من جذر. نقدم في معالجاتنا في هذا الفصل طرقاً عددية على MATLAB للوصول للحلول ذات قيم حقيقية .

(٣,١) طريقة التنصيف Bisection Method

طريقة التنصيف Bisection method هي من أبسط الطرق العددية التكرارية لإيجاد جذر معادلة، وتحتاج إلى نقطتين ابتدائيتين. تعتمد أساساً على نظرية القيمة

المتوسطة وهي التي تضمن وجود جذر في الفترة التي تتغير فيها إشارة الدالة، بتكرار تنصيف الفترات التي تحتوي على الجذر يتم التقريب للجذر.

لتكن $f(x)$ دالة متصلة على الفترة $[a,b]$ بحيث إن $f(a)$ و $f(b)$ مختلفتان في الإشارة، أي $f(a)\,f(b) < 0$ فإنه يوجد على الأقل عدد $c \in [a,b]$ بحيث $f(c) = 0$. نأخذ في البداية الفترة $[a_1,b_1]$ بحيث إن $f(a_1)$ و $f(b_1)$ مختلفتان في الإشارة، ثم نحسب $c_1 = (b_1+a_1)/2$ ونحسب $f(c_1)$. بعدها نقوم باختيار الفترة $[a_1,c]$ أو $[c, b_1]$ بحيث تكون $f(a_1)$ و $f(c_1)$ أو $f(b_1)$ و $f(c_1)$ مختلفتين في الإشارة. نكرر العملية حتى التوصل للجذر أو لجذر بالتقريب المطلوب. نحاول حصر جذر واحد فقط في الفترة المنصفة. وإذا كانت الفترة التي تحتوي على الجذر غير معلومة، نستطيع الاستعانة بالرسم *plot* لتحديد تلك الفترة. نخزن في MATLAB m-file والذي يحتوي على الخوارزمية (٣,١) *bisect* [7]، ويتم استخدام البرنامج بمناداته مع تحديد المعادلة، فترة التنصيف والدقة المطلوبة.

مثال رقم (٣,١)

أوجد جذر المعادلة $x^3-2x-1=0$ في الفترة [1.5, 2] بطريقة التنصيف .

الحل :

نخزن المعادلة في m-file يدعى fun :

```
function f=fun(x)
f=x.^3-2*x+ -1;
```

```
function solution=bisect(fun,a,b,acc)
fa=feval(fun,a);
fb=feval(fun,b);
while abs(b-a)>acc
   c=(a+b)/2;
   fc=feval(fun,c);
   if fa*fc<=0;
      b=c;
   else a=c;
   end
end
solution=(a+b)/2;
```

خوارزمية (٣،١)

ثم ندخل في نافذة الأوامر :

```
>>bisect('fun',1.5,2,1e-2)

solution =
   1.6211
```

وتظهر النتيجة بقيمة الجذر. طريقة التنصيف بسيطة و دائمة التقارب ولكن من عيوبها البطء في الوصول للجذر.

(٣،٢) طريقة نيوتن Newton Method

طريقة نيوتن Newton method هي من أشهر وأقوى الطرق التكرارية لإيجاد جذور المعادلة $f(x) = 0$. وتحتاج طريقة نيوتن لفرض نقطة بداية x_0 وتولّد مع التكرار المتتالية x_n التي تتقارب إلى الجذر الفعلي. إذا كانت النقطة الابتدائية قريبةً بما فيه الكفاية من الجذر p، والجذر بسيطاً، (جذر غير مكرر) أي أن تكون مشتقة الدالة عند الجذر p قيمة غير صفرية $(f'(p) \neq 0)$ فإن طريقة نيوتن تتقارب بسرعة. تحتاج الطريقة

لقيم الدالة *f(x)* و المشتقة الأولى *f '(x)* ، وتعتمد هندسياً على المماس لـ *f(x)*. تدعى الطريقة أيضاً بطريقة نيوتن– رافسون Newton-Raphson method.

إذا كانت *f(x)* دالة متصلة وقابلة للاشتقاق على [a,b] و $f'(x) \neq 0$ لكل $x \in [a,b]$ ، فإن طريقة نيوتن تولّد متتالية $\{x_n\}$ المعرّفة بالتالي :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad n \geq 1.$$

نبرمج الطريقة باسم *newton* ونخزنها في m-file [15] (خوارزمية ٣,٢) ويتطلب الأمر تحديد الدالة ، والمشتقة ، ونقطة البداية ويمكن تحديد قيمة الدقة المطلوبة tol بحيث يتم التوقف عن حساب قيم جديدة في المتتالية x_n إذا :

$$\left| \frac{f(x_n)}{f'(x_n)} \right| < \text{tol}$$

```
function[r,it]=newton(fun,dfun,x,acc)
it=0;
x0=x;
d=feval(fun,x0)/feval(dfun,x0);
while abs(d)>acc
   x1=x0-d;
   it=it+1;
   x0=x1;
   d=feval(fun,x0)/feval(dfun,x0);
end;
r=x0;
```

خوارزمية (٣,٢) .

مثال رقم (٣,٢)

استخدم البرنامج newton المعطى في خوارزمية (٣,٢) لإيجاد جذر المعادلة $x^3-10x^2+29x-20=0$ بنقطة البداية $x_0=7$.

الحل :

نخزن الدالة و المشتقة في m-files باسم *fun* و *dfun* :

```
function f=fun(x)
f=x.^3-10*x.^2+29*x-20;
```

```
function f=dfun(x)
f=3*x.^2-20*x+29;
```

نطبّق الأمر newton مع تحديد الدالة ، والمشتقة ، و نقطة البداية ، والدقة المطلوبة ، ليعطي نتيجة الحل مع عدد خطوات التكرار :

```
>>[r,it]=newton('fun','dfun',7,.00005)
r =
   5.0000
it =
    6
```

ويمكن تعديل الطريقة إذا كان هناك صعوبة في المشتقة بالتعويض عن المشتقة بقيمة الميل ، وتصبح :

$$x_n = x_{n-1} - \frac{(x_{n-1} - x_{n-2})f(x_{n-1})}{f(x_{n-1}) - f(x_{n-2})} \quad .$$

تسمى هذه الطريقة بطريقة القاطع Secant method وتحتاج إلى نقطتي بداية ،

وتوجد نسخة من البرنامج في الملحق [7] ، عند استخدام برنامج القاطع *secant* على نفس المثال و بنقطتي بداية 4 و 4.5 وبنفس الدقة نحصل على :

```
>>sol=secant('fun',4,4.5,.00005)
sol=
     5.0000
it=
   8
```

التقارب بهذه الطريقة أبطأ من طريقة نيوتن ، و كلا الطريقتين بطيئة في حال كانت الجذور قريبة جداً من بعضها أو إذا كانت جذوراً مضاعفة (تتكرر أكثر من مرة واحدة كجذر للمعادلة). إذا كانت الجذور مضاعفة فيمكن معالجة الأمر وتسريع التقارب باستخدام طريقة نيوتن المعدلة Modified Newton method ومعادلتها :

$$x_{n+1} = x_n - \frac{f(x_n)f'(x_n)}{f'(x_n)^2 - f(x_n)f''(x_n)}$$

العائق الوحيد في طريقة نيوتن المعدلة هو حساب المشتقة الثانية $f''(x)$ وازدياد كمية العمليات الحسابية. يستطيع القارئ استخدام البرنامج الموجود في الملحق [7] باسم *modifiedNewton* :

مثال رقم (٣,٣)

احسب جذر المعادلة :

$$f(x) = (x-1)^2 \ln(x)$$

نلاحظ أن الجذر p=1 مكرر، فنطبق برنامج طريقة نيوتن المعدلة على

المعادلـة ، وهـو يحتـاج إلى إدخـال الدالـة ، والمـشتقة الأولى ، والمـشتقة الثانيـة ، ونقطـة البداية ، و الدقة المطلوبة.

```
>> [sol,it]=modifiedNewton('fun','dfun','ddfun',.9,.00005)
sol =
   1.0000
it =
    3
```

للمقارنـة ، نـستخدم برنـامج نيـوتن علـى نفـس الدالـة ، وبـنفس نقطـة البدايـة والدقـة. نلاحـظ أن طريقـة نيـوتن احتاجـت إلى 17 خطـوة ولم تـصل إلى نتيجـة دقيقـة. وبـذلك نـستنتج أن طريقـة نيـوتن تحتـاج إلى خطـوات أكثـر لتـصل للجـذر بـسبب كـون الجذر مكرراً ، بينما طريقة نيوتن المعدلة احتاجت فقط ثلاث خطوات :

```
>> [sol,it]=newton('fun','dfun',.9,.00005)
sol =
   0.9999
it =
   17
```

(٣,٣) إيجاد جذور معادلات باستخدام دوال جاهزة في MATLAB

يقـدم MATLAB دوال جاهزة مختلفـة تـساعد في إيجـاد حلـول لمعـادلات خطيـة وغير خطية منها :

(٣,٣,١) دالة fzero

الدالـة الجاهـزة *fzero* تقـوم بحـساب جـذور المعـادلات ، وهـي عبـارة عـن توليفـة من الطرق العددية التي تعطي نتائج مضمونة. ويتم استخدام الأمر بعد تعريف الدالـة *f*

في *inline* أو m-file ثم تحديد الفترة التي تحتوي على الجذر والدقة المطلوبة ، والتي يمكن الحصول عليها برسم الدالة بالأمر *plot* :

```
>> f=inline('x^3-2*x^2-x+2')
f =
    Inline function:
    f(x) = x^3-2*x^2-x+2
>>sol=fzero(f,[0 1.5], 1e-15)
sol =
    1
```

إذا لم يكن للدالة جذرٌ فعلي محدد جبرياً مثل الدالة *cos(x)-x* فيمكن أن نحصل على تقريب للجذر بالأمر *fzero* :

```
>> function z=f(x)
    z = cos(x)-x;
>> format long
>> sol=fzero('f',[0 2],1e-15)
sol =
   0.73908513321516
```

يمكن استخدام الأمر *fzero* بإدخال الدالة ونقطة البداية فقط ، ولكن قد لا يصل للجذر خاصة إذا كان الجذر قريباً من نقطة غير معرفة بالنسبة للدالة. وإذا لم يتم تحديد الدقة المطلوبة فإن MATLAB يحسب بالدقة $2x10^{-16}$.

```
>> f=inline('sin(x)-.5*x')
f =
    Inline function:
    f(x) = sin(x)-.5*x
>> sol=fzero(f,1)
sol =
   1.89549426703398
```

(٣,٣,٢) دالة roots

توجد دالة أخرى جاهزة في MATLAB تستخدم لإيجاد أصفار كثيرات الحدود تدعى *roots (c)* ، حيث إن المتجه *c* هو متجه معاملات كثيرة الحدود.

مثال رقم (٣,٤)

اوجد جـــــذور $x^3-2x^2-x+2=0$ بالدالة *roots* حيـــث متجه المعامـــلات : c=[1 -2 -1 2]

```
>> roots([1 -2 -1 2])
ans =
  - 1.0000
    2.0000
    1.0000
```

(٣,٣,٣) دالة solve

نستطيع استخدام دالة solve في الجبر الرمزي *Symbolic algebra* لحل المعادلة السابقة (كثيرة الحدود) كالآتي :

```
>> syms x
>> sol=solve(x^3-2*x^2-x+2)
 sol =
 -1
  1
  2
```

أو لتقريب حل للمعادلة غير الخطية $cos(x)-x=0$:

```
>> syms x
>> sol=solve(cos(x)-x)
 sol =
0.73908513321516
```

(٣,٤) حل نظام معادلات غير الخطية

إذا كان لدينا نظام من المعادلات غير الخطية فيمكن حلها باستخدام طريقة تعتمد على طريقة نيوتن. نبدأ بفرض وجود معادلتين غير خطيتين في متغيرين :

$$f_1(x,y) = 0$$
$$f_2(x,y) = 0$$

بحيث إن $f_1(x,y)$ و $f_2(x,y)$ دوال متصلة وتمثل النظام بالمعادلة $\mathbf{F(x)}=0$ حيث إن $F(x,y)=(f_1(x,y), f_2(x,y))^t$ ، و نبحث عن الحل $(x,y)^t$ الذي يحقق المعادلتين.

نحتاج في طريقة نيوتن لحل هذا النظام إلى المشتقة باستخدام المصفوفة الجاكوبية Jacobian matrix :

$$J(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix}$$

لتصبح صيغة المعادلة التكرارية لحل النظام بطريقة نيوتن و بنقطة بداية $x^{(0)}$ مناسبةٍ لكل $n \geq 1$ هي :

$$\boldsymbol{x}^{(n)} = \boldsymbol{x}^{(n-1)} - \boldsymbol{J}(\boldsymbol{x}^{(n-1)})^{-1}\boldsymbol{F}(\boldsymbol{x}^{(n-1)})$$

ولتفادي حساب معكوس مصفوفة الجاكوبية فنستخدم :

$$\boldsymbol{J}(\boldsymbol{x}^{(n-1)})\boldsymbol{h}^{(n-1)} = -\boldsymbol{F}(\boldsymbol{x}^{(n-1)})$$
$$\boldsymbol{x}^{(n)} = \boldsymbol{x}^{(n-1)} + \boldsymbol{h}^{(n-1)}$$

وهي الصيغة التكرارية المستخدمة في الخوارزمية (٣،٣) لطريقة نيوتن لنظام معادلات غير خطية التي تدعى [15] *newton2* . وعند استخدامها تتم المناداة بالمدخلات : نقطة البداية المناسبة (x,y) ، والمعادلات ، والمشتقات ، وعدد المتغيرات ، والدقة المطلوبة.

```
function [xx,it]=newton2(x,f,jf,n,tol)
it=0;
xx=x;
fr=feval(f,xx);
while norm(fr)>tol
   jr=feval(jf,xx);
   xx1=xx-jr\fr;
xx=xx1;
   fr=feval(f,xx);
   it=it+1;
end
```

خوارزمية (٣،٣)

مثال رقم (٣،٥)

في نظام معادلات غير خطية كالآتي :

$$x^2 + y^2 = 16$$

$$xy = 1$$

الحل يمثل بنقاط التقاطع المبينة في الشكل رقم (٣.١) الذي حصلنا عليه بالأوامر التالية :

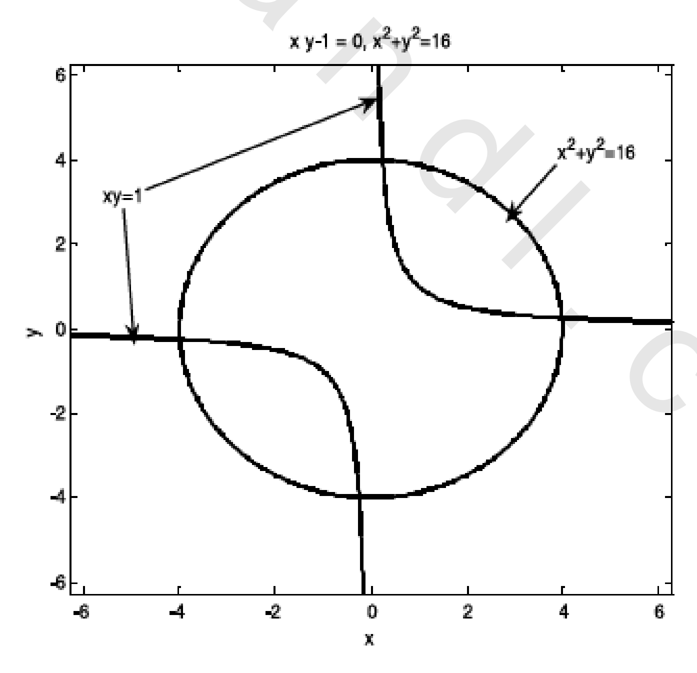
```
>> ezplot('x^2 + y^2 - 16')
>> hold
Current plot held
>> ezplot('x*y-1')
```

تخزن المعادلات في m-file يدعى *f1* :

```
function f=f1(v)
x=v(1);y=v(2);
f=zeros(2,1);
f(1)=x^2+y^2-16;
f(2)=x*y-1;
```

والمشتقة الجاكوبية في m-file آخر يدعى *f2* :

```
function jf=f2(v)
 x=v(1);y=v(2);
jf=zeros(2,2);
 jf(1,:)=[x*2 y*2];
 jf(2,:)=[y x];
```



الشكل رقم (٣,١). نقاط التقاطع لنظام المعادلات غير خطية (مثال رقم ٣,٥).

نستخدم برنامج newton 2 مع تحديد نقطة البداية المناسبة (x,y)، المعادلات من ملف f1 ، المشتقات من ملف f2 ، عدد المتغيرات 2 ، والدقة 0.00005 :

```
>> [sol1,iter]=newton2([4 1]', 'f1','f2',2,0.00005)
sol1 =
   3.9921
   0.2505
iter =
    3
```

ونحصل على أحد الجذور (0.2505, 3.9921) بعد ثلاثة تكرارات. وباختيار نقاط بداية مختلفة نحصل عليها من الرسم ونستطيع الحصول على باقي الجذور الثلاثة :

```
>> [sol2,iter]=newton2([.5 3]', 'f1','f2',2,0.00005)
Sol2 =
   0.2505
   3.9921
iter =
    4

>> [sol3,iter]=newton2([-4 0]', 'f1','f2',2,0.00005)
Sol3 =
  -3.9921
  -0.2505
iter =
    3
>> [sol4,iter]=newton2([0 -4]', 'f1','f2',2,0.00005)
Sol4 =
  -0.2505
  -3.9921
iter =
    3
```

مثال رقم (٦,٣)

أوجد حلاً للنظام غير الخطي التالي :

$$\sin x + y^2 + \ln z = 7$$
$$3x + 2y - z^3 = -1$$
$$x + y + z = 5$$

الحل :

نخزن المعادلات في m-file يدعى *f101* و المشتقة الجاكوبية في m-file آخر يدعى *f201* :

```
function q=f101(p)
x=p(1);y=p(2);z=p(3);
q=zeros(3,1);
q(1)=sin(x)+y^2+log(z)-7;
q(2)=x*3+2^y-z^3+1;
q(3)=x+y+z-5;

function jq=f201(p)
x=p(1);y=p(2);z=p(3);
jq=zeros(3,3);
jq(1,:)=[cos(x) 2*y 1/z];
jq(2,:)=[3 (2^y )*log(2) -3*(z^2)];
jq(3,:)=[1 1 1 ];
```

نستخدم برنامج newton 2 مع تحديد نقطة البداية المناسبة (0,2,2)، وعدد المتغيرات ثلاثة، والدقة 0.001 . لنحصل على الجذر(*2.005 , 2.3959 , 0.5991*) وبعد ثلاثة تكرارات :

```
[x,it]=newton2([0,2,2]','f101','f201',3,.001)
x =
   0.5991
   2.3959
   2.0050
it =
    3
```

(٣,٥) تمارين

١– استخدم طريقة التنصيف لإيجاد جذر المعادلة e^x-2-x على الفترة [-3, -2] وبالدقة 10^{-4} .

٢– استخدم طريقة التنصيف لإيجاد جذر المعادلات بالدقة 10^{-4} :

أ) $x-2^{-x}=0 \quad 0\le x \le 1$

ب) $x^4-2x^3-4x^2+4x+4=0 \quad -1\le x\le 0\,,\ 2\le x\le 3,\ 0\le x\le 2,\ -2\le x \le -1$

٣– استخدم طريقة نيوتن لإيجاد جذور المعادلات في التمرين رقم ٢.

٤– استخدم طريقة نيوتن لإيجاد جذر المعادلة x^3-3x+1 على الفترة [1, 3] ونقطة بداية $x_0=1.5$.

٥– أوجد قيمة تقريبية بطريقة نيوتن للجذر $\sqrt{7}$ بنقطة بداية 2.5.

٦– أوجد نقطة التقاطع للدالتين $sin(x)\ ,\ e^{-x}$ في [0,1].

٧– استخدم طريقة رباعية التقارب لإيجاد الجذر $x=0$ للمعادلة $e^x-x-1=0$.

٨– استخدم طريقة القاطع secant method لإيجاد الجذر للمعادلة $f(x)=-x^3-cos(x)$ واستخدم طريقة التنصيف لإيجاد نقطتي البداية.

٩– استخدم طريقة القاطع secant method لإيجاد الجذر الموجب للمعادلة $x^{10}-1\ =0$ بنقطتي بداية $x_0=1.2\ x_1=1.1$.

١٠– استخدم طريقة نيوتن المعدلة لإيجاد جذر المعادلة المكرر $x\ lnx - lnx$ عند $p=1$.

١١– حل النظام التالي بطريقة نيوتن بنقطة بداية (1,1) و بدقة 10^{-7} :

$$4x^3+y=6$$
$$x^2y=1$$

١٢– استخدم طريقة نيوتن بنقطة بداية $x^{(0)}=0$ لإيجاد $x^{(2)}$ للنظام غير الخطي :

$$4x_1^2 - 20x_1 + \frac{1}{4}x_2^2 + 8 = 0$$
$$\frac{1}{2}x_1x_2^2 + 2x_1 - 5x_2 + 8 = 0$$

١٣- أوجد نقطتي التقاطع بين المعادلتين وارسم للتأكد من موقع التقاطع بدقة 10^{-7}.

$$-x_1(x_1 + 1) + 2x_2 = 18$$
$$(x_1 - 1)^2 + (x_2^2 - 6)^2 = 25$$

الفصل الرابع

حساب التفاضل والتكامل في MATLAB

نقدم في هذا الفصل طرقاً عددية لتقريب أهم مبادئ حساب التفاضل والتكامل Calculus ونعرض إمكانات MATLAB في تبسيط هذه الطرق العددية التي تعتمد على تعريف الدوال في هيئة جداول بيانية متقطعة، وذلك ربما لعدم وجود صيغة محددة للدالة أو لصعوبة الاشتقاق أو التكامل بالطرق المعتادة. من مزايا برنامج MATLAB أنه يستطيع التعامل مع البيانات مهما كان حجمها كبيراً وبطرق دقيقة جداً. بالاستعانة بالإمكانات القوية للرسم على MATLAB نستطيع إعطاء هذه البيانات تمثيلاً بيانياً لتسهيل تحليلها واستخراج خواصها مثل الاتصال، والقيمة العظمى والصغرى، وتحديد قابليتها للاشتقاق والتكامل، وغيرها من الخواص. كما سنعرض مواضيع في حساب التفاضل والتكامل في عدة متغيرات في الجزء الأخير من الفصل.

يعّد مفهوم النهاية من المفاهيم الأساسية في الرياضيات وعلى وجه الخصوص في حساب التفاضل والتكامل. فالمسائل الأساسية في علم التفاضل والتكامل مثل إيجاد المشتقة عند نقطة أو إيجاد المساحة تحت منحنى دالة ما، تتركز حول مفهوم النهاية. وسوف نقدم قدرات MATLAB على رسم وحساب النهايات.

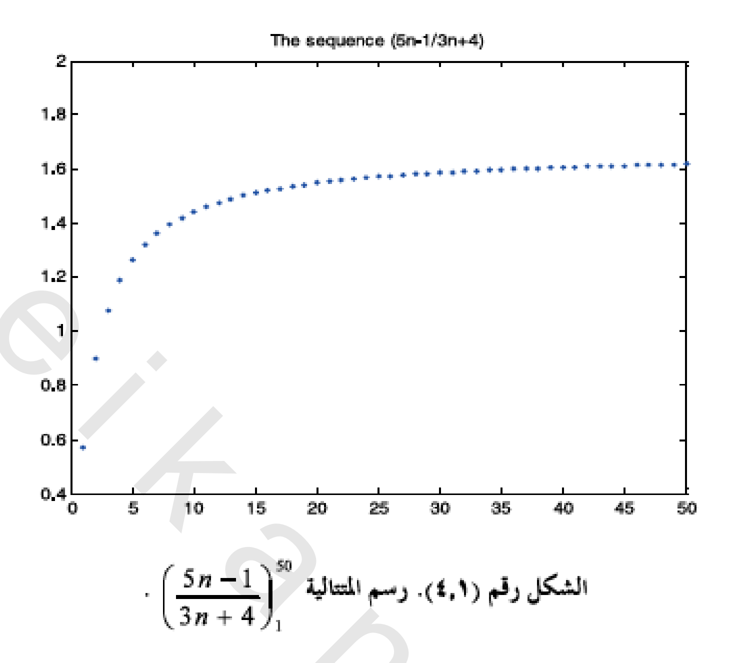
(٤,١) المتتاليات والمتسلسلات Sequences and Series

(٤,١,١) المتتاليات

المتتالية غير المنتهية Infinite sequence هي دالة مجالها الأعداد الكلية ومداها الأعداد الحقيقية أو الأعداد المركبة وتكتب $(a_n)_1^\infty$. مع أن MATLAB لا يتعامل مع المتجهات غير المنتهية لكن يمكننا أن نختار عدداً كبيراً جداً من الحدود يصل إلى 10^{300} وهذا يكفي لنلاحظ الصورة العامة للمتتالية في الرسم. أما في البيئة الرمزية Symbolic Algebra فيمكن التعامل مع المتتاليات غير المنتهية و يمكننا حساب النهايات إن وجدت.

التمثيل بالرسم يعطي الشكل العام للمتتالية وطريقة تقاربها إن وجد (الشكل رقم ٤,١). ونقوم برسم المتتاليات على شكل نقاط بعد تحديد عدد الحدود n *plot(a_n,'.')*. فعند رسم المتتالية وتحديد n بالقيمة 50 $\left(\dfrac{5n-1}{3n+4}\right)_1^\infty$ نلاحظ من الرسم أن المتتالية تتقارب من 1.6 ويمكن حساب النهاية بالأمر *limit* في *syms* :

```
>> n=1:50
>> a=(5*n-1)./(3*n+4);
>> plot(a,'.')
>> syms n
>> limit((5*n-1)/(3*n+4),n,inf)
 ans =
5/3
```



الشكل رقم (٤،١). رسم المتتالية $\left(\frac{5n-1}{3n+4}\right)_1^{50}$.

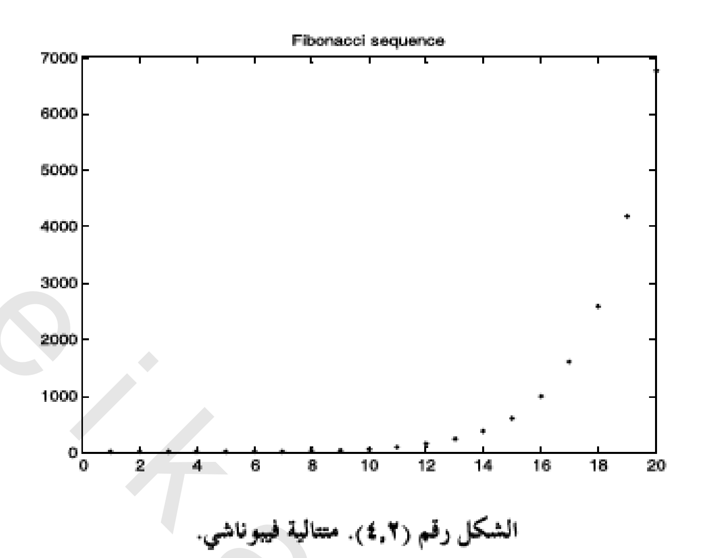
مثال رقم (٤،١)

إحدى أكثر المتتاليات رواجاً في التطبيقات الرياضية هي متتالية فيبوناشي Fibonacci sequence وتعرّف كالتالي :

$$f_1 = f_2 = 1,$$
$$f_n = f_{n-2} + f_{n-1} \quad n = 3,4,...$$

ويمكن رسمها (الشكل رقم ٤،٢) بالأوامر التالية ، مع تحديد n=20 :

```
>> f=[1 1];
>> for n=3:20
f=[f  f(n-2)+f(n-1)];
end
>> plot(f,'.')
```



الشكل رقم (٤,٢). متتالية فيبوناشي.

(٤,١,٢) المتسلسلات

مفهوم مرتبط بالمتسلسلة غير المنتهية infinite series $\sum_{1}^{\infty} a_n$ هو متتالية للمجاميع الجزئية $(s_n)_1^{\infty}$ sequence of partial sums ، بحيث $s_n = \sum_{k=1}^{n} a_k$ ويوجد في MATLAB الأمر *cumsum* الذي يقوم بحساب هذه المجاميع. فمثلا للمتسلسلة $\sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{4^n}$ يمكننا تعريف متتالية المجاميع الجزئية $s_n = \sum_{k=1}^{n} \frac{(-1)^{k-1}}{4^k}$ كما يمكن حساب أول ١٠٠ مجموع ورسم متتالية المجاميع بالأوامر التالية :

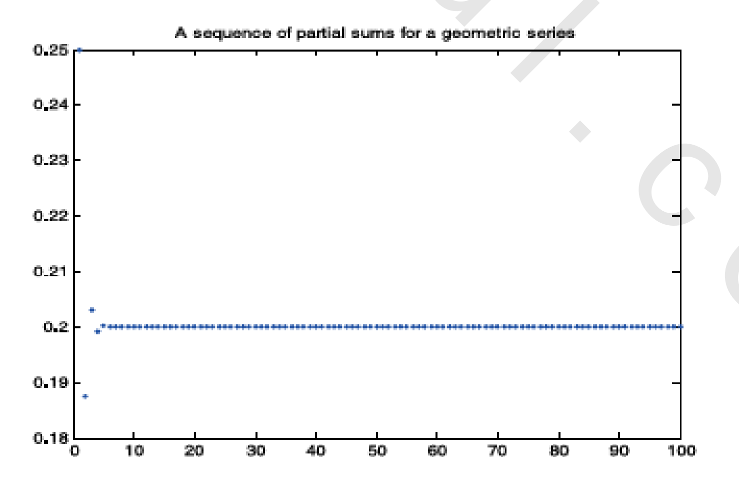
```
>> n=1:100;
>> a=(-1).^(n-1)./4.^n;
>> s=cumsum(a);
>> plot(s,'.')
```

من الشكل رقم (٤,٣). نستنتج أن المتسلسلة هي متسلسلة هندسية geometric series وتتقارب إلى 0.2 . في بيئة الحساب الرمزي يمكننا حساب مجموع المتسلسلة الهندسية العامة الذي $\sum_{0}^{\infty} r^n$ يساوي $\frac{1}{1-r}$ إذا $|r| \prec 1$ باستخدام الأمر *symsum* :

```
>> syms r n
>> s=symsum(r^n,n,0,inf)
s =
-1/(r-1)
```

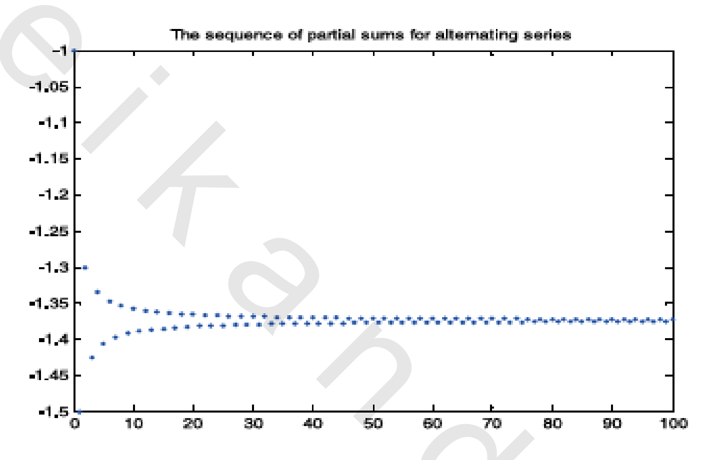
وفي المثال السابق الحد الابتدائي *a = 1/4* و *r = -1/4* ، ليصبح المجموع :

```
>> s=symsum((-1)^(n-1)/4^n,n,1,inf);
>> double(s)
ans =
   0.2000
```



الشكل رقم (٤,٣). متتالية المجاميع $\left(\sum_{k=1}^{n} \frac{(-1)^{k-1}}{4^k}\right)_1^{100} = (s_n)_1^{100}$.

هناك نوع آخر من المتسلسلات هو المتسلسلة المتذبذبة alternating series مثل $\sum_{n=0}^{\infty}\frac{(-1)^n}{3n-1}$ ، ويمكن رسم المجاميع الجزئية المئة الأولى ونلاحظ من الشكل رقم (٤،٤). أنها متقاربة.



الشكل رقم (٤،٤). متتالية المجاميع لمتسلسلة متذبذبة .

(٤،٢) التفاضل العددي Numerical Differentiation

بصورة عامة المشتقة للدالة $f(x)$ عند النقطة x تُعرف :

$$f'(x) \equiv \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

إن وجدت النهاية. والمشتقة العددية تعتمد على إيجاد قيمة تقريبية للكسر عند h ذات القيمة الصغيرة و المناسبة. التفاضل العددي يُعد من الطرق غير المستقرة لتراكم اخطاء التدوير عند تصغير المقدار h وبذلك لن يعطي تقريباً أفضل بمجرد اختيار h صغيرة ،

ولكن يجب البحث عن المقدار h الصغير والذي يحافظ على استقرار الطريقة. علماً أننا نعدّ الطرق العددية مستقرة stable إذا أجرينا تغيرات بسيطة في الشروط الابتدائية، مما يؤدي إلى تغيرات بسيطة في النتائج النهائية.

(٤,٢,١) الفروق الجزئية Divided differences

الفروق الجزئية divided differences هي إحدى الطرائق العددية المستخدمة في تقريب الدوال بكثيرات الحدود، ولها عدة درجات فالفرق الجزئي الصفري للدالة f بالنسبة لـ x_i هو $f[x_i] = f(x_i)$ ، أي قيمة f عند x_i ، والفرق الجزئي الأول بالنسبة لـ x_i و x_{i+1} (حيث إن $x_{i+1} = x_i + h$) هو:

$$f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

أي قيمة تقريبية للنهاية المستخدمة في تعريف مشتقة الدالة f عند النقطة x ، وبذلك تساعد الفروق الجزئية في تقريب المشتقة . الأمر *diff* في MATLAB يعطي الفروق الجزئية للمتجه، فمثلاً :

```
x = [1 2 7 9 10];
>> y = diff(x)
y =
   1    5    2    1
```

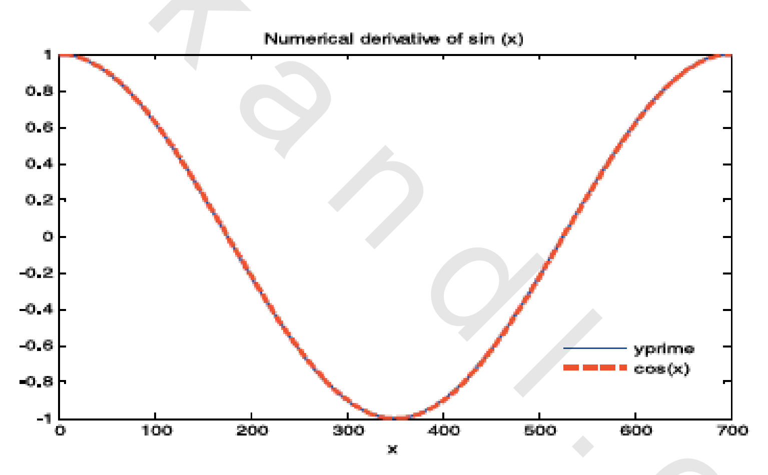
يعطينا الفروق الجزئية للمتجه (1 2 7 9 10) مع ملاحظة أن المتجه الناتج أقصر من المتجه الأصلي.

مثال رقم (٤,٢)

لحساب المشتقة الأولى للدالة *sin(x)* نفرض أن لدينا متجهاً من 700 حد ، ونحسب بالفروق الجزئية قيمة تقريبية للمشتقة باستخدام *yprime= diff(y)./diff(x)*

ونقارن النتائج بالمشتقة الأصلية *cos (x)* . في الرسم الموضح في الشكل رقم (٤,٥) يظهر تطابق المتجهين *cos (x)* من المشتقة الأصلية والمتجه *yprime* من الفروق التجزيئية.

```
>> x=linspace(0,2*pi,700);
>> yprime=diff(y)./diff(x);
>> plot(cos(x),':r')
>> hold
Current plot held
>> plot(yprime)
>> hold off
```



الشكل رقم (٤,٥). التفاضل العددي للدالة sin(x) .

الطريقة السابقة لتقريب المشتقة هي من أبسط الطرق وتستعمل فقط نقطتين(x_0 , x_0+h)، ولكن هناك طرقاً أخرى تستعمل ثلاث نقاط أو أكثر لتعطي حلولاً أدق. وهذه الطرق تُشتق من متسلسلة تايلور أو من كثيرة الحدود لاجرانج Lagrange polynomials التي سيتم عرضها بالتفصيل في فصل الاستكمال.

وتُعد كثيرات الحدود من أشهر الدوال وأكثرها استخداماً، خصوصاً في

التقريب والاستكمال، لأنها دوال متصلة، ولسهولة حساب كل من مشتقاتها وتكاملاتها. الموضوع الذي سنتوسع بعرض تطبيقاته في الفصل السادس. من كثيرات الحدود التي تستخدم في التقريب متسلسلة تايلور، وهي طريقة أساسية في التحليل العددي لتقريب الدوال، وتُعرف كثيرة الحدود تايلور $p_n(x)$ من الدرجة n للدالة f حول x_0 بالتالي :

بفرض f دالة متصلة و جميع المشتقات $f^{(n+1)}$ متصلة على الفترة [a,b] ($f\in C^n[a,b]$) و $x_0 \in [a,b]$ فلكل $x\in[a,b]$ يوجد $\zeta(x)$ بين x_0 و x يحقق $f(x)=P_n(x)+R_n(x)$ حيث إن :

$$p_n(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^n(x_0)}{n!}(x-x_0)^n=\sum_{k=0}^{n}\frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$$

$$R_n(x)=\frac{f^{(n+1)}(\xi(x))}{n+1!}(x-x_0)^{n+1}$$

ويسمى $R_n(x)$ الحد الباقي (او خطأ القطع) truncation error المرافق لـ $p_n(x)$. ويطلق على المتسلسلة اللانهائية التي نحصل عليها بأخذ نهاية $p_n(x)$ عندما n تؤول إلى مالانهاية بمتسلسلة تايلور للدالة f حول x_0 .

والمفهوم العام لخطأ القطع هو الخطأ الناتج عن استخدام مجموع مقطوع أو مجموع منته كمجموع لمتسلسلة لانهائية. و يوفر MATLAB في البيئة الرمزية *syms* إمكانية كتابة متسلسلة تايلور Taylor series لدالة ما وحول نقطة محددة. فمثلاً لإيجاد متسلسلة تايلور من الدرجة الخامسة للدالة $f(x) = \sin x$ وحول $a = \pi$.

```
>> syms x
>> t5=taylor(sin(x),pi,6)
t5 =
-x+pi+1/6*(x-pi)^3-1/120*(x-pi)^5
```

للحصول على طرق عددية لتقريب المشتقة نحتاج ثلاث نقاط أو أكثر لنشر الدالة f بواسطة كثيرة الحدود تايلور من الدرجة n عند x_0 ومن ثم حساب كثيرة الحدود عند نقاط مختلفة (x_0+h أو x_0-h , ...) . وبعمليات جبرية بسيطة مثل الجمع أو الطرح لهذه المعادلات الناتجة يمكن الوصول للصيغ المختلفة. كما أن طريقة اختيارالنقاط يعطي طرقاً مختلفة لتقريب المشتقة، فمثلاً قانون الفروق الأمامية forward differences approximation نحصل عليه باختيار النقاط الثلاث الأمامية x_0 , x_0+h , x_0+2h ومقدار الخطوة $h > 0$:

$$f'(x_0) = \frac{1}{2h}[-3f(x_0) + 4f(x_0+h) - f(x_0+2h)] + \frac{h^2}{3} f^{(3)}(\xi_0)$$

و يدعى قانون الفــروق الخلفية backward differences approximation في حالة $h < 0$.

كما يوجد قانون الفروق الوسطية central differences approximation حين تتوسط نقطة الاشتقاق النقطتين x_0+h و x_0-h :

$$f'(x_0) = \frac{1}{2h}[f(x_0+h) - f(x_0-h)] - \frac{h^2}{6} f^{(3)}(\xi_1)$$

كما توجد طرق عددية لتقريب المشتقة الثانية :

$$f''(x_0) = \frac{1}{h^2}[f(x_0-h) - 2f(x_0) + f(x_0+h)] - \frac{h^2}{12} f^{(4)}(\xi)$$

ونلاحظ في هذه الصيغ من الحد الأخير (حد الخطأ) أن الخطأ يتناسب طردياً مع h^2 ، أو يمكننا القول إن جميع هذه الصيغ هي بخطأ برتبة $O(h^2)$.

ويمكن الحصول على تقريب لمشتقات أعلى ، وذلك باستخدام متسلسلة تيلور بدرجات مختلفة وعند نقاط مختلفة. وقد تم اختيار صيغ مختلفة لإيجاد المشتقات الأربع الأولى وجمعها في برنامج واحد باسم *diffgen* [15] (ويمكن للقارئ حساب صيغ أخرى وبرمجتها في الخوارزمية حسب احتياجه) وهي :

$$f'(x_0) \approx \frac{1}{12h}[f(x_0-2h)-8f(x_0-h)+8f(x_0+h)-f(x_0+2h)]$$

$$f''(x_0) \approx \frac{1}{12h^2}[-f(x_0-2h)+16f(x_0-h)-30f(x_0)+16f(x_0+h)-f(x_0+2h)]$$

$$f'''(x_0) \approx \frac{1}{8h^3}[f(x_0-3h)-8f(x_0-2h)+13f(x_0-h)-13f(x_0+h)+8f(x_0+2h)-f(x_0+3h)]$$

$$f^{(iv)}(x_0) \approx \frac{1}{6h^4}[-f(x_0-3h)+12f(x_0-2h)-39f(x_0-h)+56f(x_0)-39f(x_0+h) +12f(x_0+2h)-f(x_0+3h)]$$

عند استخدام الخوارزمية (٤.١) نحصل على قيم المشتقات الأربع الأولى لأي دالة عند النقطة المحددة من المستخدم ، علما أنها كلها صيغ ذات خطأ برتبة $O(h^4)$.

```
function v = diffgen(fun,n,x,h)
if ((n==1)|(n==2)|(n==3)|(n==4))
  c=zeros(4,7);
0];    c(1,:)=[  0  1  -8  0   8  -1
0]; 1    -   c(2,:)=[  0  -1  16  -30  16
  c(3,:)=[1.5  -12  19.5  0  -19.5  12  -1.5];
  c(4,:)=[  -2  24  -78  112  -78 24  -2];
  p=feval(fun,x+[-3:3]*h);
  v=c(n,:)*p';
  v=v/(12*h^n);
end
```

خوارزمية (٤,١).

مثال رقم (٤,٣)

استخدم البرنامج *diffgen* المعطى في خوارزمية (٤.١) وذلك لإيجاد المشتقات الأربع الأولى للدالة $y=x^{11}$ عند :

$x = 1$ و لقيم h المتناقصة من 0.05 إلى $5x10^{-5}$.

الحل :

تم تخزين الدالة في الملف *f400*، و بتكرار الأمر *diffgen('f400',n,1,h)* تظهر النتيجة في الجدول رقم (٤.١) :

الجدول رقم (٤,١). نتائج مثال (٤,٣) .

h	1st derivative	2nd derivative	3rd derivative	4th derivative
0.05000	10.98835	109.97680	989.39027	7918.78457
0.00500	11.00000	110.00000	989.99994	7919.99989
0.00050	11.00000	110.00000	990.00001	7919.94855
0.00005	11.00000	110.00000	989.98409	6448.17533

ونلاحـــظ أن النتائج بدأت تتغـــير، والدقة بدأت تتناقص عند أصغر قيمـــة $h = 5x10^{-5}$ وذلك بسبب أخطاء التدوير الناتجة من بعض الصيغ .

(٤,٢,٢) دالة diff الرمزية

في بيئة الحسابات الرمزية Symbolic Algebra يوفر MATLAB أوامر لحساب

المشتقات بدرجات مختلفة، ونستطيع استخدام الأمر *diff (f)* في *syms* لإيجاد المشتقة الأولى للدالة *f* وللحصول على المشتقة *n* ندخل *diff(f,n)* .

مثال رقم (٤,٤)

إذا كان لدينا الدالة $f = x/(1+x^2)$ فيتم حساب المشتقة الأولى بالأوامر التالية :

```
>> syms x
>> f=x/(1+x^2);
>> fderiv=diff(f)
fderiv =
1/(1+x^2)-2*x^2/(1+x^2)^2
```

ونحسب المشتقة الثانية باستخدام *diff(f,2)* :

```
>> fderiv2=diff(f,2)
fderiv2 =
-6/(1+x^2)^2*x+8*x^3/(1+x^2)^3
```

أما إذا كانت الدالة في أكثر من متغير، فيجب تحديد المتغير المستقل المراد حساب المشتقة بالنسبة إليه.

مثال رقم (٤,٥)

احسب المشتقة الثالثة بالنسبة للمتغير *x* للدالة $f = x\,y/(1+x^2)$.

الحل :

بالأمر *diff(f,'x',3)* نحصل على المطلوب :

```
>> syms x y
>> f=x*y/(1+x^2);
>> fderiv3x=diff(f,'x',3)
fderiv3x =
 48*y/(1+x^2)^3*x^2-6*y/(1+x^2)^2-48*x^4*y/(1+x^2)^4
```

هناك تطبيقات عديدة على الاشتقاق في علم التفاضل ، ونقدم فيما يلي بعض الأمثلة على ذلك.

مثال رقم (٦, ٤)

أوجد نقاط الانقلاب للدالة $f(x)=\dfrac{3x^2-2}{x^3}$.

الحل :

نستخدم الأوامر *diff(f,'x',1)* و *diff(f,'x',2)* لحساب المشتقتين الأولى والثانية للدالة :

```
>> f=(3*x^2-2)/x^3;
>> fderiv1=diff(f,'x',1)
fderiv1 =
6/x^2-3*(3*x^2-2)/x^4
 >> fderiv2=diff(f,'x',2)
fderiv2 =
-30/x^3+12*(3*x^2-2)/x^5
```

ولإيجاد جذور المشتقة الثانية نستخدم الأمر *solve(fderiv2)* :

```
 >> solve(fderiv2)
ans =
  2
 -2
```

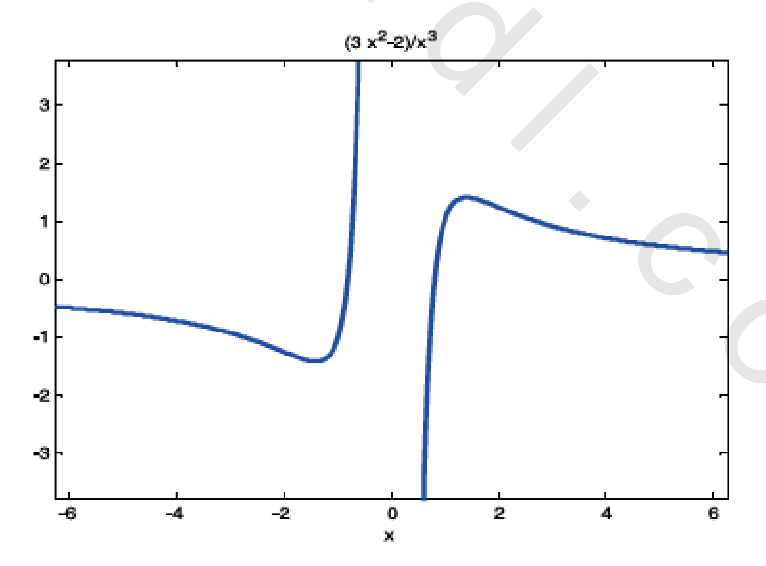
فنحصل على الجذرين 2 و 2- ونحسب *f(2)*, *f(-2)* باستخدام *feval(f, 2)* و *feval(f, -2)* :

```
>> f=inline(f)
f =
    Inline function:
    f(x) = (3.*x.^2-2)./x.^3


>> feval(f,2)
ans =
   1.2500


>> feval(f,-2)
ans =
  -1.2500
```

نستنتج أن الدالة مقعرة لأعلى في كل من (2,∞) و (-2,0) ، ومقعرة لأسفل في كل من (-∞,-2) و (0,2) . ومن ثم فإن النقطتين (2,1.25) و (-2,-1.25) ، تشكلان نقطتي انقلاب للدالة ، كما هو موضح في الشكل رقم (٤,٦).



الشكل رقم (٤,٦). رسم للدالة في مثال رقم (٤,٦).

مثال رقم (٤,٧)

أوجد القيم القصوى المحلية للدالة $f(x)=3x^4-8x^3+6x^2-1$.

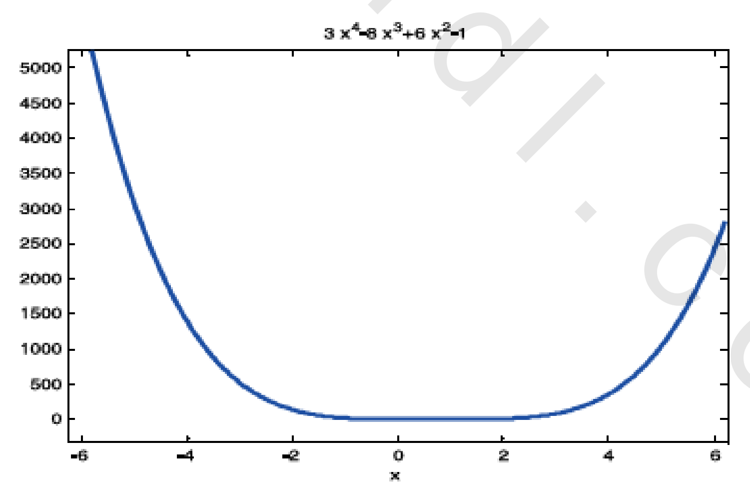
الحل :

نبدأ الحل بالرسم (الشكل رقم ٤,٧) بدالة *ezplot* :

```
>> syms x
>> f=3*x^4-8*x^3+6*x^2   -1;
>> ezplot(f)
```

ونحسب المشتقة الأولى للدالة *f* :

```
>> fderiv1=diff(f,'x',1)
fderiv1 =
12*x^3-24*x^2+12*x
```



الشكل رقم (٤,٧). رسم للدالة في مثال رقم (٤,٧).

نحصل على أصفار المشتقة الأولى بالأمر :

```
>> solve(fderiv1)
ans =
 0
 1
 1
```

وبذلك تصبح 0 و 1 هما النقطتان الحرجتان، ونستخدم اختبار المشتقة الثانية لتحديد القيم القصوى المحلية :

```
>> fderiv2=diff(f,'x',2)
 fderiv2 =
 36*x^2-48*x+12

>> df2=inline(fderiv2)
df2=
    Inline function:
    df2(x) = 36.*x.^2-48.*x+12
```

ونحسب $f''(0)$:

```
>> feval(df2,0)
ans =
   12
```

ومن ثم فإن $f(0) = 1$ هي قيمة صغرى محلية. ولكن $f''(1) = 0$ و بذلك فإن اختبار المشتقة الثانية يفشل في تصنيف النقطة 1 ولكن اختبار المشتقة الأولى والرسم البياني يبينان أن $f(1)$ ليست قيمة قصوى محلية.

مثال رقم (٤,٨)

أوجد المستقيمات المقاربة الأفقية و الرأسية للدالة :

$$f(x) = \frac{3x}{\sqrt{x^2 + x}}$$

الحل :

نحسب النهايات للدالة عند ∞ و ∞− :

```
>> limit((3*x)/sqrt(x^2+x),x,inf)
ans =
3
>> limit((3*x)/sqrt(x^2+x),x,-inf)
 ans =
 -3
```

وهذا يعني أن $y = 3$ مستقيم مقارب أفقي عند ∞ و $y = -3$ مستقيم مقارب أفقي عند ∞ - .

بما أن أصفار المقام تساعد في إيجاد المستقيمات المقاربة الرأسية ، و مجال الدالة هو$(0,\infty) \cup (-\infty,-1)$ فنكتفي بحساب النهايات التالية :

```
>> limit((3*x)/sqrt(x^2+x),x,0)
 ans =
0
 >> limit((3*x)/sqrt(x^2+x),x,-1,'left')
 ans =
 -Inf
```

وبذلك يصبح $x = -1$ هو المستقيم المقارب الرأسي الوحيد.

(٤,٣) التكامل Integration

(٤,٣,١) مجموع ريمان Riemann Summation

يتم تقريب المساحة تحت منحنى الدالة $y = f(x)$ وعلى الفترة [a,b] بالمجموع :

$$R_n = \sum_{k=1}^{n} f(x_k)\Delta x$$

حيث إن عدد الفترات n هو عدد موجب، وطول كل شريحة هو $\Delta x=(b-a)/n$ ، والنقاط $x_i = a + i\Delta x$ لكل i=0,1,...,n ، باستخدام تجزيء منتظم للفترة [a,b] و x_k هي نقطة اختيارية في الفترة الجزئية i . يسمى R_n بمجموع ريمان Riemann Sum والمساحة تحت المنحنى للدالة *f(x)* تعرّف بنهاية هذا المجموع (إن وجدت) في حال أن n تؤول إلى ما لانهاية $n \to \infty$. عندما توجد النهاية فإن المساحة تحت منحنى الدالة *f(x) ما* بين a و b تعرف بالتكامل المحدود Definite Integral ويرمز له :

$$\int_a^b f(x)dx$$

مثال رقم (٤,٩)

لحساب مجموع ريمان للدالة *cos(x)* على الفترة [0, π] وعند النقطة الاختيارية المحددة بمنتصف الفترة، فإن المجموع يصبح بتحديد n =100 ، a = 0 و b = π :

```
>> deltax=(pi-0)/100
deltax =
   0.0314
```

```
>> x=deltax/2:deltax:pi-deltax/2;
>> rn=sum(cos(x))*deltax
rn =
  2.4415e-017
```

(٤,٣,٢) التكامل العددي Numerical Integration

يوجد تكامل فعلي للعديد من الدوال، ولكن لبعض الدوال من الصعب حسابه، إما لعدم وجود دالة أصلية واضحة أو أنه ليس من السهل الحصول على الدالة

الأصلية، وفي هذه الحالات نستخدم طريقة عددية لتقدير التكامل. برنامج MATLAB يُعد من أقوى الأدوات التي تقرب عددياً التكامل المحدود لوجود دوال جاهزة أنشئت لذلك مع امكانية برمجة الطرق العددية المعروفة للتكامل العددي بسهولة.

(٤,٣,٢,١) قاعدة سمبسون المركبة Composite Simpson Rule

من أكثر الطرق العددية انتشاراً لحساب التكامل وتقديره هي طريقة سمبسون المركبة Composite Simpson's Rule . فإذا كان لدينا الدالة $y = f(x)$ معرّفة على الفترة [a,b] وعرفنا تجزيئاً منتظماً على n فترات جزئية بحيث إن $n = 2m$ عدد زوجي ، و $h=(b-a)/n$ هو طول كل فترة ، فإن طريقة سمبسون المركبة تقدم تقريباً للتكامل $\int_a^b f(x)dx$ بالمجموع :

$$S_n = \frac{h}{3}\left(f(a) + 4\sum_{k=1}^{n/2} f(x_{2k-1}) + 2\sum_{k=1}^{n/2-1} f(x_{2k}) + f(b)\right)$$

حيث إن $x_k = a+kh$ و $k = 0,1,\ldots,n$.

ويقوم برنامج *simpsons* بحساب التكامل بطريقة سمبسون المركبة [15] في الخوارزمية (٤,٢):

```
function s=simpsons(fun,a,b,n)
h=(b-a)/n;
x=[a:h:b]; y=feval(fun,x);
v=2*ones(n+1,1);
v2=2*ones(n/2,1);
v(2:2:n)=v(2:2:n)+v2;
v(1)=1; v(n+1)=1;
s=y*v;
s=s*h
```

خوارزمية (٤,٢)

مثال رقم (٤,١٠)

اســــتخدم برنامـــج *simpsons* في حســـــاب قيمة تقريبيـــة للتكامل $\int_0^{\pi}\cos(x)dx = 0$ بـ n=40 .

الحل :

نادي البرنامج لنحصل على النتيجة التقريبية للصفر :

```
>> sn=simpsons('cos',0,pi,40)
sn =
 1.3545e-016
```

(٤,٣,٢,٢) قاعدة شبه المنحرف المركبة Composite Trapezoidal Rule

طريقـة عدديـة أخـرى لحساب التكامـل هـي طريقـة شـبه المنحـرف المركبـة Composite Trapezoidal Rule التي تقسّم المنطقة تحت المنحنى إلى أجزاء تأخذ شكل شبه المنحرف، ولتقريب التكامل $\int_a^b f(x)dx$ تستخدم القانون:

$$T_n = \frac{h}{2}(f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b))$$

حيث إن $h = \frac{b-a}{n}$ و $x_i = a+ih$ لكل $i = 0,1,...n$.

البرنامج *trapezoidal* المعطى في الخوارزمية (٤,٣) [7] يحتوي على طريقة شبه المنحرف، وهو يحتاج إلى تعريف الدالة المراد حساب تكاملها في m-file باسم fun والحدود a وb و عدد الفترات n .

```
function tp=trapezoidal(fun,a,b,n)
h=(b-a)/n;
t=(feval(fun,a)+feval(fun,b))/2;
for k=1:n-1
   x=a+h*k;
   t=t+feval(fun,x);
end
tp=t*h;
```

خوارزمية (٤,٣).

للمقارنة قمنا باستخدام برنامج *trapezoidal* لنفس المثال عند n=100 كالآتي :

```
>> tp=trapezoidal('fun',0,pi,100)
tp =
 -1.3951e-016
```

قمنا باستخدام عدد فترات أكثر في طريقة شبه المنحرف ، وذلك بتحديد n = 100 لتحسين دقة الحل ، ولكن كما نلاحظ فإن طريقة سمبسون المركبة تتفوق. ويرجع ذلك التحسن لكون طريقة سمبسون المركبة تستخدم كثيرات حدود لاغرانج من الدرجة الثانية في تقريب الدالة المراد إيجاد تكاملها وهذا يحسن الخطأ بصورة ملحوظة إذ إن الخطأ يكون برتبة $O(h^4)$ ، بينما قاعدة شبه المنحرف المركبة تستخدم كثيرة حدود لاغرانج الخطية ، وهناك يكون الخطأ برتبة $O(h^2)$.

(٤,٣,٣) دوال MATLAB الجاهزة للتكامل العددي

يمكن برمجة كل الطرق العددية في m-file ومن ثم استخدامها مثل طريقة Gaussian Quadrature ، ولكن MATLAB يوفر دوال جاهزة للتكامل تدعى *quadl*

quad و *quad8* . الأمر *quad* يستخدم طريقة سمبسون المركبة لإيجاد التكامل . والأمر *quad8*يستخدم طريقة نيوتن كوتس الثامنة Adaptive Recursive Newton Cotes ، وفي *quadl* يستخدم تربيع جاوس- لوباتو adaptive Gauss/Lobatto qudrature rule. وفي كل تلك الأوامر الجاهزة للتكامل العددي يتوجب على المستخدم تسمية الدالة ، وحدود التكامل ، والدقة المطلوبة.

مثال رقم (٤,١١)

أوجد قيمة تقريبية للتكامل $\int_0^2 e^{-x^2}\,dx$ بخطأ أقل من 0.0001 .

الحل :

ندخل الخطوات التالية لتعريف الدالة $exp(-x^2)$ في الملف *fun* :

```
>> quad('fun',0,2,.0001)
ans =
   0.8821
>> quad8('fun',0,2,.0001)
ans =
   0.8821
```

يمكننا أيضاً حساب تكامل لدالة معرّفة بالأمر @ ومن ثم تطبيق تكامل تربيع جاوس بالحدود المطلوبة :

```
>> F = @(x) 1./(x.^3-2*x-5);
Q = quadl(F,0,2);
>> Q = quadl(F,0,2)
Q =
  -0.4605
```

(٤,٣,٤) دالة *int* الرمزية

توجد طريقة أخرى في MATLAB لحساب التكامل وهي باستخدام البيئة الرمزية *syms* و الأمر *int* كما في المثال (٤,١٢).

مثال رقم (٤,١٢)

نحســـب التكامل $\int_{1/2}^{1} \frac{x}{1+x^2} dx$ بالأمر *int* بعد تعريـــف المتغيرات x, y متغيرات رمزية حقيقية:

```
>> syms x y real
>> f=x/(1+x^2)
f =
 x/(1+x^2)
```

عند كتابة الأمر نحتاج إلى إدخال الدالة، وحدود التكامل:

```
>> b=int(f,0.5,1)
b =
3/2*log(2)-1/2*log(5)
```

نستخدم الأمر *pretty* لكتابة النتائج بطريقة منسقة وسهلة القراءة، أما الأمر *double* فلعرض النتائج على شكل عدد عشري:

```
>> pretty(b)
                3/2 log(2) - 1/2 log(5)
>> double(b)
ans =
   0.2350
```

في حال احتواء التكامل على أكثر من متغير، فيمكن حساب التكامل بالنسبة للمتغير المطلوب بتحديد ذلك في الأمر *int* فلحساب $\int_{1}^{5}\frac{y}{1+x^2}dx$ ندخل :

```
>> f=y/(1+x^2)
f =
 y/(1+x^2)
 >> b=int(f,'x',1,5)
b =
atan(5)*y-1/4*pi*y
```

أما إذا كانت حدود التكامل معطاة بدلالة a و b مثل $\int_{a}^{b}\frac{\sqrt{x}}{1+x}dx$ فيمكن حساب التكامل بنفس الطريقة وتظهر النتيجة بدلالة a و b :

```
>> syms x  a b real
>> f=(x^.5)/(1+x)
f =
 x^(1/2)/(1+x)
>> b=int(f,'x',a,b)
 b =
 2*b^(1/2)-2*atan(b^(1/2))-2*a^(1/2)+2*atan(a^(1/2))
```

وفي حال التكامل غير المحدود $\int x^3\cos(x)dx$ فيمكن حسابه بالأمر *int* دون إدخال قيم لحدود التكامل :

```
>> g=int(x^3*cos(x))
 g =
 x^3*sin(x)+3*x^2*cos(x)-6*cos(x)-6*x*sin(x)+C
```

أما التكامل على فترة غير منتهية فيمكن حسابه بتحديد ذلك في الحدود، فمثلاً التكامل $\int_0^\infty \exp(-x^2)dx$ يتم حسابه بالخطوات التالية :

```
>> syms x
>> int(exp(-x^2),0,inf)
ans =
1/2*pi^(1/2)
```

(٤,٤) تطبيقات على التكامل

حساب التكامل يحتوي على العديد من التطبيقات التي تظهر في مجالات مختلفة من العلوم والهندسة، نقدم بعض هذه التطبيقات الرئيسة في الأجزاء التالية .

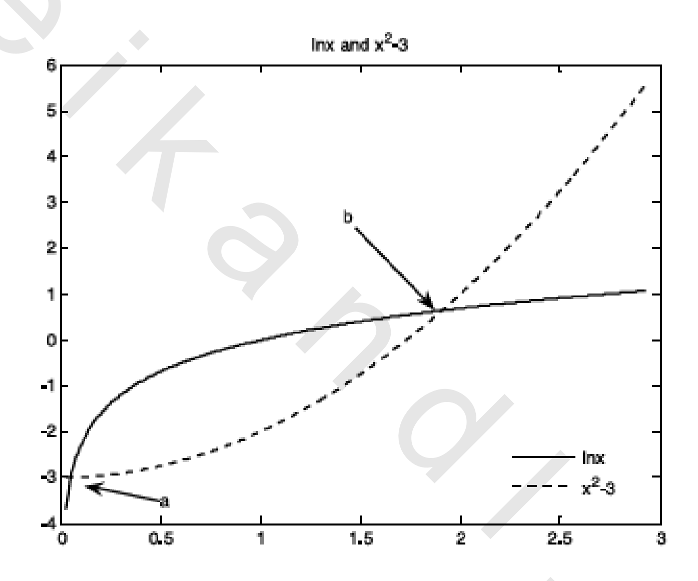
(٤,٤,١) مساحة مناطق محدودة بمنحنيات Area between curves

أحد التطبيقات الشائعة على التكامل هو حساب المساحة المحصورة ببياني دالتين متصلتين $f(x)$ و $g(x)$ ويتم استخراج صيغة التكامل $\int_a^b f(x)-g(x)dx$ من الرسم حيث يكون بيان الدالة $f(x)$ أعلى من بيان الدالة $g(x)$ وحدود التكامل يتم إيجادها من حدود المنطقة المطلوبة. ويتطلب أحياناً حساب نقاط التقاطع بين الدالتين.

المنطقة المحصورة بين بياني الدالتين $g(x) = x^2-3$ و $f(x) = ln(x)$ يمكن حساب مساحتها بالتكامل $\int_a^b \ln x - x^2 + 3dx$. ويتم حساب حدود التكامل a و b من نقاط التقاطع عن طريق حل المعادلة $x^2-3 = ln(x)$ وذلك باستخدام دالة *fzero* كما يمكن استخدام الرسم للتعرف على المنطقة المحصورة بين المنحنيين :

```
>> b=fzero('x.^2-3-log(x)',1.5)
b =1.9097
>> a=fzero('x.^2-3-log(x)',[.002 .5])
a =
   0.0499
>> x=a/2:.01:b+a/2;
>> plot(x,log(x),x,x.^2-3)
```

ليظهر لنا الشكل رقم (٤،٨):



الشكل رقم (٤،٨). المساحة المحصورة بين منحنيين.

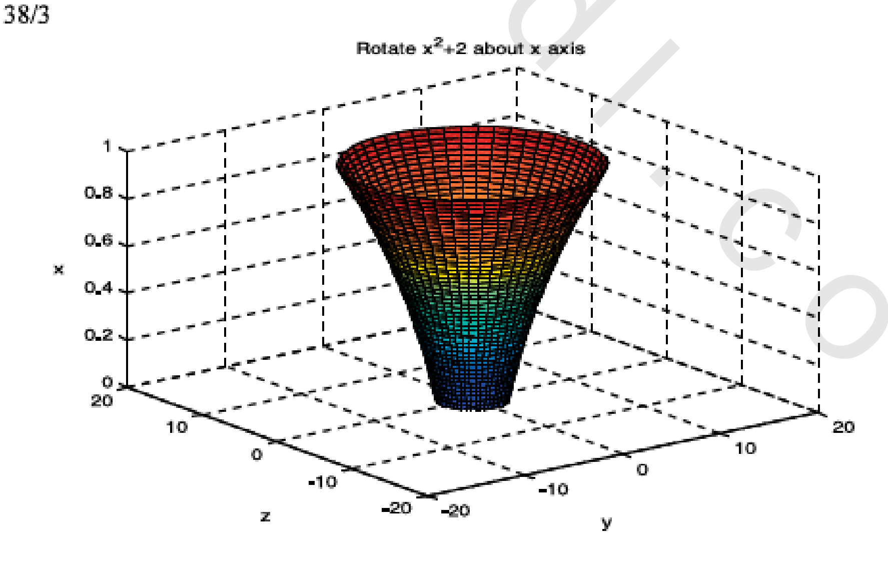
بإجراء التكامل على syms نحصل على المساحة المطلوبة:

```
>> syms x
>> A=int(log(x)-x^2+3,a,b);
>> double(A)
ans =
   2.7832
```

(٤،٤،٢) حجوم الأجسام الدورانية Solids of Revolution

حساب الحجم الناتج عن دوران منحنى دالة حول أحد المحورين *x* أو *y* يتطلب حساب التكامل. مثلاً لرسم الحجم الناتج عن دوران $y = x^2+2$ والمعرفة على الفترة [1,3] حول المحور *x* نستخدم الأمر *cylinder(y,50)* حيث 50 هي عدد النقاط على المنحنى، للحصول على مصفوفة النقاط ، نستخدم الأمر *cylinder(y,50)* حيث 50 هي عدد النقاط على المنحنى، للحصول على مصفوفة النقاط ، والأمر *surf* ينتج الرسم المطلوب (الشكل رقم ٤،٩)، ولكن دائماً أمر *cylinder* يعرّف محور الدوران بقياس من 0 إلى 1 . ثم نحسب حجم الجسم الدوراني بالأمر *int* :

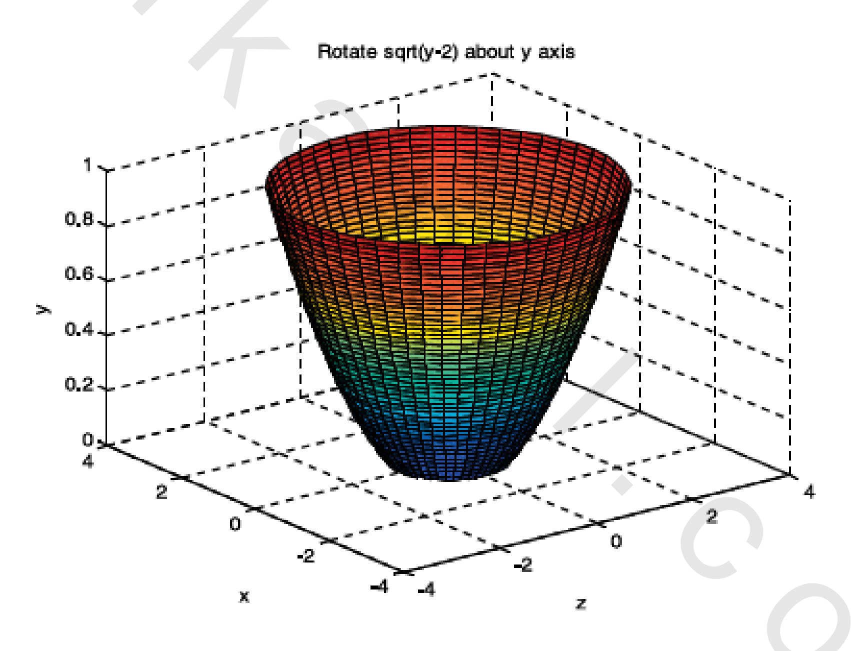
```
>> x=linspace(1,3,50);
>> y=x.^2+2;
>> [d,e,f]=cylinder(y,50);
>> surf(d,e,f);
>> syms x
>> int(x.^2+2,1,3)
ans =
38/3
```



الشكل رقم (٤،٩). الحجم الناتج عن دوران $y=x^2+2$ حول محور x .

وإذا كـان الـدوران حـول المحـور y فإننـا نـستخدم الأمـر *cylinder(x,50)* ونكتـب الدالـة بدلالـة المتغير y لتـصبح $x=(y-2)^{1/2}$ معرفـة علـى الفـترة [3,11] ونحـسب قيمـة التكامل بالأمر *int* لنحصل على الحجم المطلوب (الشكل رقم ٤,١٠) :

```
>> y=linspace(3,11,50);
>> xx=sqrt(y-2);
>> [a b c]=cylinder(xx,50);
>> surf(a,b,c);
>> int(sqrt(y-2),3,11)
ans = 52/3
```



الشكل رقم (٤,١٠). الحجم الناتج عن دوران (y-2)^.5 حول محور y .

(٤,٤,٣) طول القوس Arc length

طـول منحنى الدالـة Arc length $y = y(x)$ علـى الفـترة $a \le x \le b$ يُعطى بالقاعدة :

$$L = \int_a^b \sqrt{1+\left(\frac{dy}{dx}\right)^2}\,dx$$

وفي معظم الحالات تستعمل طريقة تقريبية لحساب التكامل لصعوبة إيجاد التكامل الفعلي.

مثال رقم (٤,١٣)

لإيجاد طول قوس منحنى الدالة :

$y = x^3 + 3x^2 - 5x + 1$ على الفترة $-1 \le x \le 2$

نستعمل تقريب للتفاضل بالأمر diff y/diffx ثم نستخدم الأمر *fcnchk* في الأمر *quad* وذلك لأن الدالة لم تكتب في m-file. وبذلك الأمر *quad* يعطي طول القوس المطلوب:

```
>> x=linspace(-1,2,400);
>>y=x.^3+3*x.^2-5*x+1;

>>dy=diff(y);
>>dx=diff(x);

>>ds=sqrt(dx.^2+dy.^2);
>> quad(fcnchk('sqrt(1+(3*x.^2+6*x-5).^2)'),-1,2,.0001)
ans =
   20.8314
```

(٤,٤,٤) مساحة سطح الدوران Surface of Revolution

يُنشأ سطح الدوران surface of revolution من دوران منحنى دالة متصلة حول مستقيم في المستوى. فإذا كان المنحنى $g(y)=x$ حيث إن g دالة ناعمة (الدالة ومشتقتها الأولى متصلة) على [c, d] ، فمساحة السطح الناتجة من دوران الجزء من المنحنى بين $y = c$ و $y = d$ حول محور x بافتراض أن $c \geq 0$ تحسب بالقاعدة :

$$S = 2\pi \int_{c}^{d} y \sqrt{1 + \left(\frac{dx}{dy}\right)^2}\, dy$$

مثال رقم (٤,١٤)

احسب مساحة السطح الناشئ عن دوران بيان الدالة $x= 2\ lny$ من $y=1$ إلـى $\sqrt{3}$ $y=$ حول محور x .

الحل :

نحسب التكامل:

$$S = 2\pi \int_{1}^{\sqrt{3}} y \sqrt{1 + \left(\frac{2}{y}\right)^2}\, dy$$

بالخطوات التالية لنحصل على السطح الناشئ عن دوران بيان الدالة :

```
>> syms y
>> s=int(y*(1+4/y^2)^(.5),1,sqrt(3))
s =
 1/2*7^(1/2)*3^(1/2)+4*log(2)-2*log(-3^(1/2)+7^(1/2))-1/2*5^(1/2)-
2*log(5^(1/2)+1)
```

```
>> S=2*pi*s
S =
 2*pi*(1/2*7^(1/2)*3^(1/2)+4*log(2)-2*log(-3^(1/2)+7^(1/2))-1/2*5^(1/2)-
2*log(5^(1/2)+1))


>> double(S)
ans =   11.1692
```

(٤,٤,٥) مساحة سطوح دوران المنحنيات الوسيطية

Area of Surfaces of Revolution of parametric curves

ليكن المنحنى C : $a \leq t \leq b$, $y = g(t)$, $x = f(t)$ منحنى ناعماً ونفرض أن C لا يقاطع نفسه، بأسلوب شبيه للسابق نجد أن مساحة سطح دوران المنحنى حول المحور x هي :

$$S = 2\pi \int_a^b g(t)\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\, dt$$

مثال رقم (٤,١٥)

احسب مساحة السطح الناشئ عن دوران المنحنى حول المحور x .

$C:\ 0 \leq t \leq \pi/3\ ,\ y=3\sin(t)\ ,\ x= 3\cos(t)$

الحل :

نحسب التكامل $S = 2\pi \int_0^{pi/3} 3\sin(t)\sqrt{9\sin^2(t) + 9\cos^2(t)}\ dt$ للحصول على مساحة السطح الناشئ عن دوران المنحنى :

```
>> syms t

>> s=int(3*sin(t)*(9*sin(t)^2+9*cos(t)^2)^(.5),0,pi/3)
 s =
 9/2

>> S=2*pi*s
S = 9*pi
```

(٤,٥) حساب التفاضل والتكامل متعدد المتغيرات Multivariable Calculus

لتعريف دالة معرفة في متغيرين مثل $f(x,y)=ye^{x^2+y^2}$ وعلى مجال مستطيل محدد في المستوى بالقيم $a\le x\le b\ ,c\le y\le d$ فيمكن كتابة m-file :

```
function z=f(x,y)
z=y.*exp(x.^2+y.^2);
```

كما نستطيع تعريف دالة في متغيرين أيضاً عن طريق الأمر *inline* :

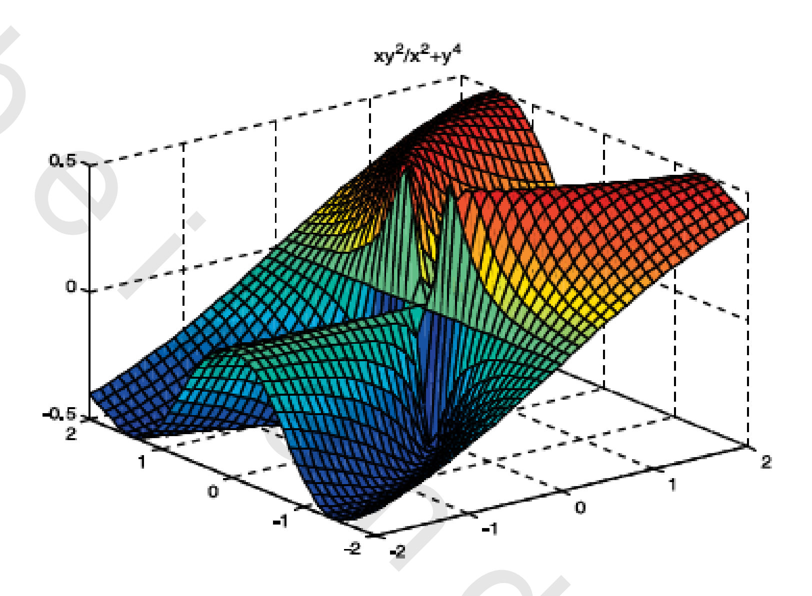
```
>> f=inline('y.*exp(x.^2+y.^2)','x','y')
f =
    Inline function:
    f(x,y) = y.*exp(x.^2+y.^2)
```

أما رسم دالة في متغيرين *z = f(x,y)* فهو رسم لسطح في الفراغ، والخطوة الأولى لرسمه هي تحديد نقاط المجال بتكوين مصفوفة بالأمر *meshgrid* ثم استخدام الأمر *surf*.

مثال رقم (٤,١٦)

الدالة $f(x,y)=\dfrac{xy^2}{x^2+y^4}$ يمكن رسمها مع أن الدالة غير متصلة عند (0,0) ولكن بإضافة eps نحصل على الشكل رقم (٤,١١):

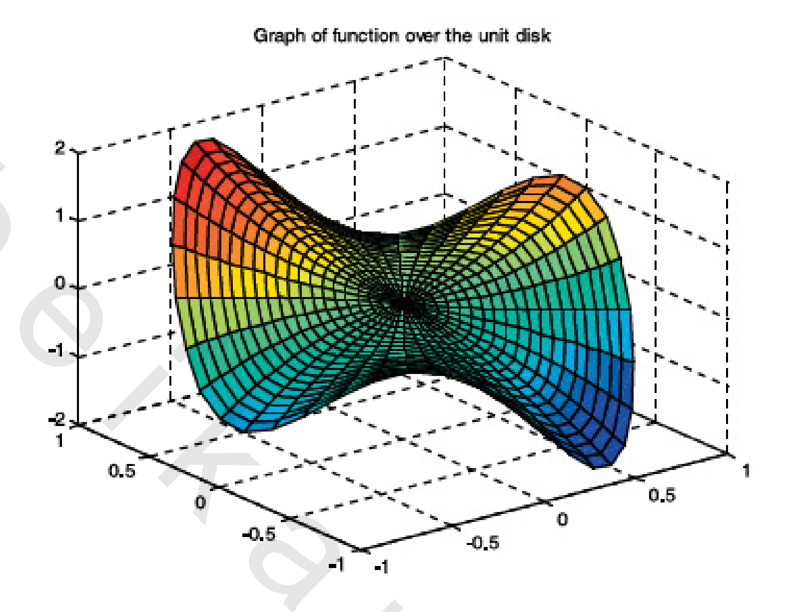
```
>> [x,y]=meshgrid(-2:.1:2);
>> surf(x,y,(x.*y.^2)./(x.^2+y.^4+eps))
```



الشكل رقم (٤,١١). دالة في متغيرين .

في بعض الحالات نريد أن نرسم دالة معرفة على مجال غير المستطيل مثلاً على قرص. ولإتمام ذلك يجب إنشاء مصفوفة المجال *meshgrid* بدلالة المحاور القطبية r , θ . فعلى سبيل المثال إدخال الخطوات التالية يتم بها رسم الدالة $g(r,\theta) = r \sin(\theta) + r^2\cos(3\theta)$ (الشكل رقم ٤.١٢)، والمعرفة على قرص الوحدة و بمركز في نقطة الأصل.

```
>> r=linspace(0,1,21);
>> theta=linspace(0,2*pi,41);
>> [rr,t]=meshgrid(r,theta);
>> x=rr.*cos(t);
>> y=rr.*sin(t);
>> z=rr.*sin(t)+rr.^2.*cos(3*t);
>> surf(x,y,z)
```

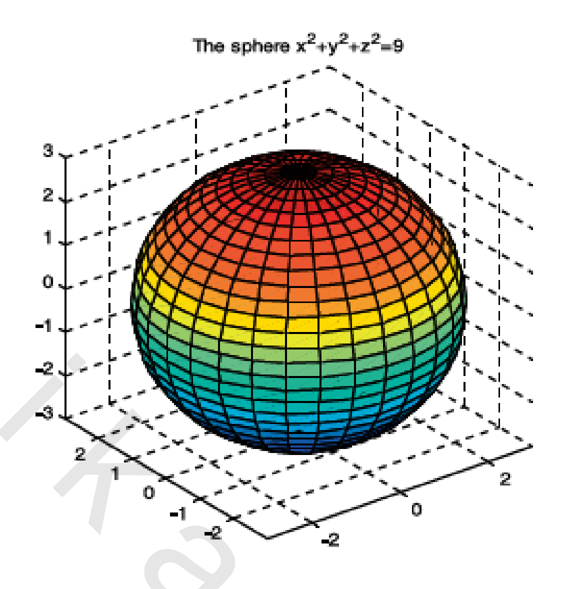


الشكل رقم (٤,١٢).دالة معرفة على قرص الوحدة.

(٤,٥,١) الأمر *sphere*

من ضمن الدوال الجاهزة في MATLAB دالة *sphere* التي تقوم بإنشاء مصفوفة النقاط للكرة مع تحديد عدد الخطوط الطولية والعرضية للسطح ، فمثلاً لرسم الكرة الممثلة بالسطح $x^2+y^2+z^2 = 9$ وبتحديد عدد الخطوط 30 نقوم بالأوامر التالية (الشكل رقم ٤,١٣) :

```
>>[x,y,z]=sphere(30);
>> surf(3*x,3*y,3*z)
```



الشكل رقم (٤,١٣). رسم الكرة.

(٤,٥,٢) رسم المتجهة المماس لمنحنى في المستوى

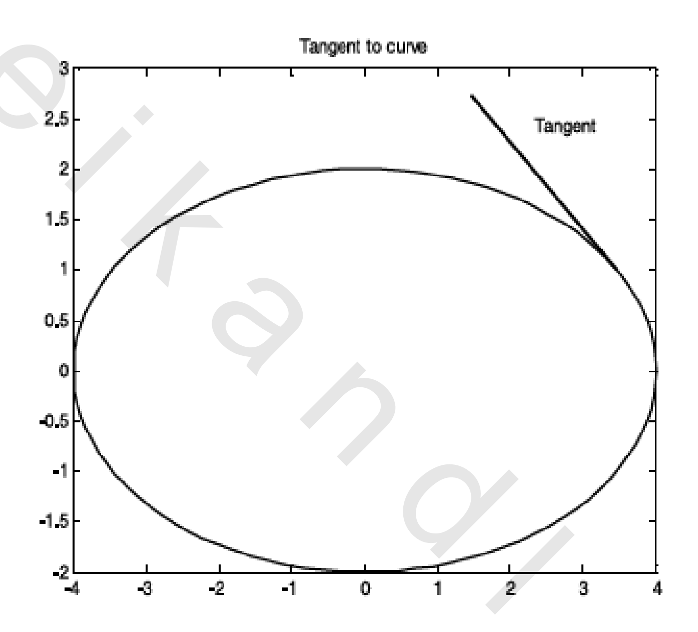
لرسم المتجهة المماس لمنحنى في المستوى Tangent to a plane curve نستخدم المحاور القطبية لرسم المنحنى *(4cost, 2sint)* ومتجه المماس *(-4sint+2cost)(π/6)* وندخل ما يلي :

```
>> t=linspace(0,2*pi);
>> x=4*cos(t);
>>y=2*sin(t);
>> xdriv=-4*sin(pi/6);
>>ydriv=2*cos(pi/6);
```

و معادلة المماس ترسم بالخطوات التالية لتنتج الشكل رقم (٤,١٤) :

```
%line parameters
```

```
>> s=[0 1];
>> v1=4*cos(pi/6)+s.*xdriv;
>> v2=2*sin(pi/6)+s.*ydriv;
>> plot(x,y,v1,v2);
```



الشكل رقم (٤،١٤). المماس لمنحنى في المستوى.

(٤,٥,٣) رسم المستوى المماس لدالة

المستوى المماس Tangent plane للدالة $z=f(x,y)$ عند النقطة $(a,b,f(a,b))$ هو رسم للتقريب الخطي Linear approximation المعطى بالمعادلة :

$$L(x,y) = f(a,b)+(x-a)f_x(a,b)+(y-b)f_y(a,b)$$

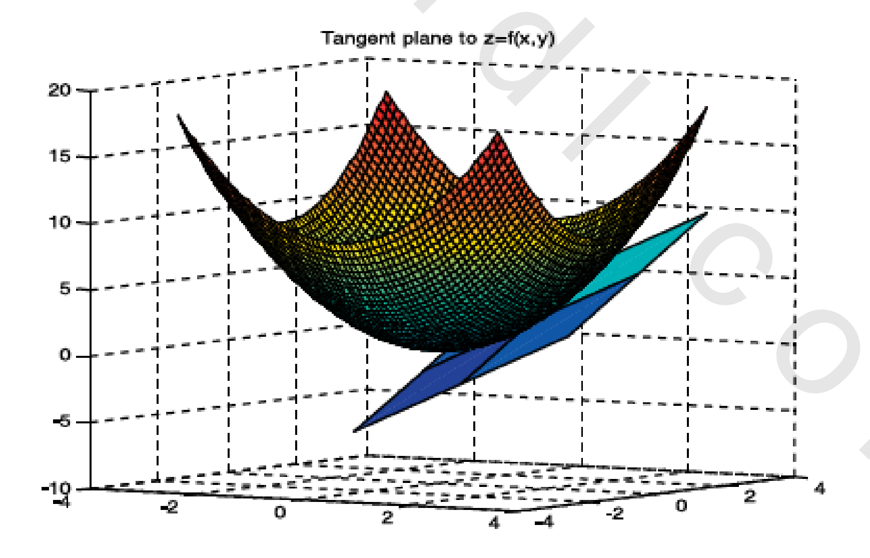
فمثلاً لرسم الدالة $z = x^2+y^2$ على الفترة $-3 \le x$ و $y \le 3$ وجزء من المستوى المماس عند النقطة (1,1,2) يصبح التقريب الخطي عند (1,1) ممثلاً بالمعادلة :

$$L(x,y)=f(1,1)+(x-1)f_x(1,1)+(y-1)f_y(1,1) = 2+2(x-1)+2(y-1)$$

ونستخدم أداة الدوران *rotate* في نافذة الرسم للحصول على أفضل زاوية لعرض الرسم (الشكل رقم ٤،١٥) ، والأمر *hold on* لعرض الدالة و المستوى المماس على نفس الرسم :

```
>>[x,y]=meshgrid(-3:.1:3);
>>surf(x,y,x.^2+y.^2);hold on
>>[u,v]=meshgrid(-2:2:2);

>>L=2+2*u+2*v;
>>surf(u+1,v+1,L);hold off
```

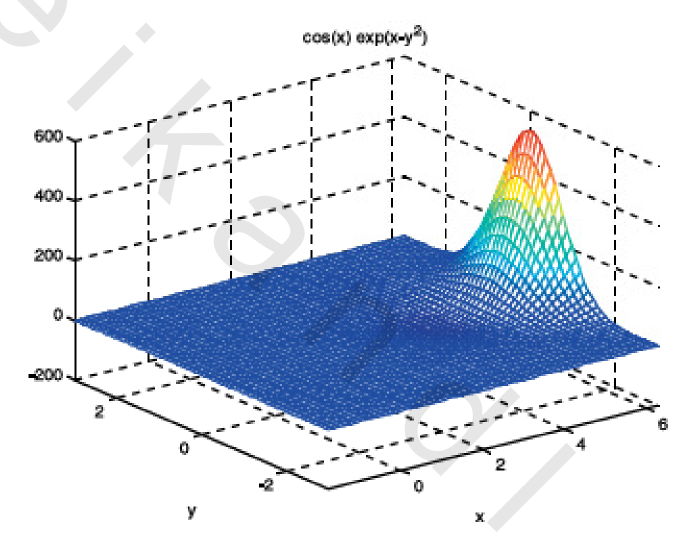


الشكل رقم (٤،١٥). المستوى المماس للدالة $z=x^2+y^2$.

(٤,٥,٤) أوامر الرسم في البيئة الرمزية

توجد أوامر في *syms* لرسم دوال في متغيرين مثل *ezsurf, ezmesh ,ezcontour*. فعلى سبيل المثال الدالة *exp(x-y²)cos(x)* ترسم بالأمر ezmesh (الشكل رقم ٤,١٦) بإدخال الأوامر التالية :

```
>> syms x y
>> f=cos(x)*exp(x-y^2);
>> ezmesh(f)
```



الشكل رقم (٤,١٦). رسم دالة باستخدام ezmesh .

نعرض في الجزء التالي بعض التطبيقات الرياضية المختلفة في حساب التفاضل والتكامل في عدة متغيرات.

(٤,٥,٥) حل نظام معادلتين في متغيرين

لإيجاد حل نظام معادلتين في متغيرين :

$$f(x,y)=0$$
$$g(x,y)=0$$

نستخدم الأمر *solve* في *syms* .

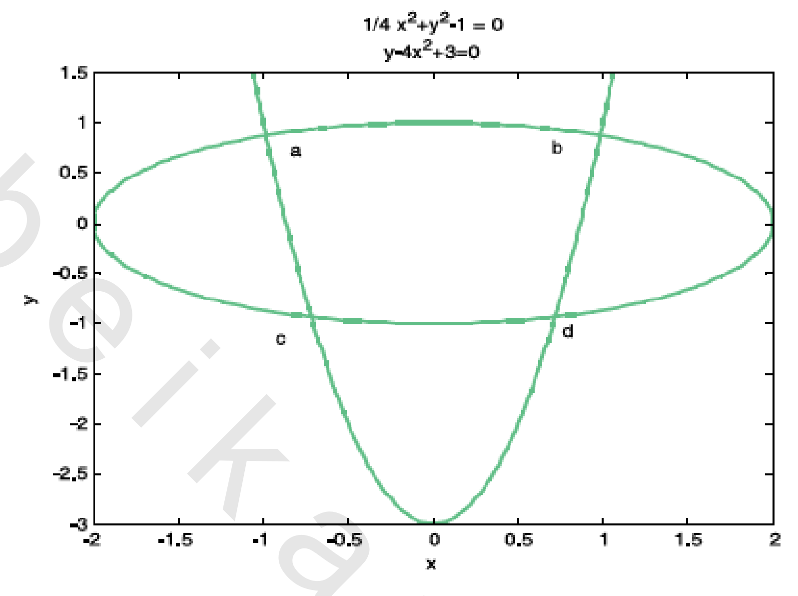
مثال رقم (٤,١٧)

نفرض أن لدينا نظاماً :

$$f(x,y) = y-4x^2+3$$
$$g(x,y) = x^2/4+y^2-1$$

عند رسم المنحنيات $f(x,y) = 0$ و $g(x,y) = 0$ نلاحظ وجود أربعة جذور عند تقاطع المنحنيين a,b,c,d [9] (الشكل رقم ٤,١٧). ولحساب الجذور الأربعة نُدخل الآتي :

```
>> syms x y
>> f=y-4*x^2+3;
>> g=.25*x^2+y^2-1;
>> [xx,yy]=solve(f,g)
xx =
 -1/16*(190+14*17^(1/2))^(1/2)
 1/16*(190+14*17^(1/2))^(1/2)
-1/16*(190-14*17^(1/2))^(1/2)
 1/16*(190-14*17^(1/2))^(1/2)
yy =
 -1/32+7/32*17^(1/2)
-1/32+7/32*17^(1/2)
-1/32-7/32*17^(1/2)
-1/32-7/32*17^(1/2)
>> double([xx yy])
ans =
  -0.9837    0.8707
   0.9837    0.8707
  -0.7188   -0.9332
   0.7188   -0.9332
```



الشكل رقم (٤,١٧). رسم نظام دالتين في متغيرين.

(٤,٥,٦) التفاضل في عدة متغيرات

حساب المشتقات الجزئية f_x, f_{xx}, f_{xy} للدالة $f(x,y) = sin(x^2+y^2)$ على syms يتم بالأمر *diff(f,'x')* مع تحديد المتغير ودرجة الاشتقاق.

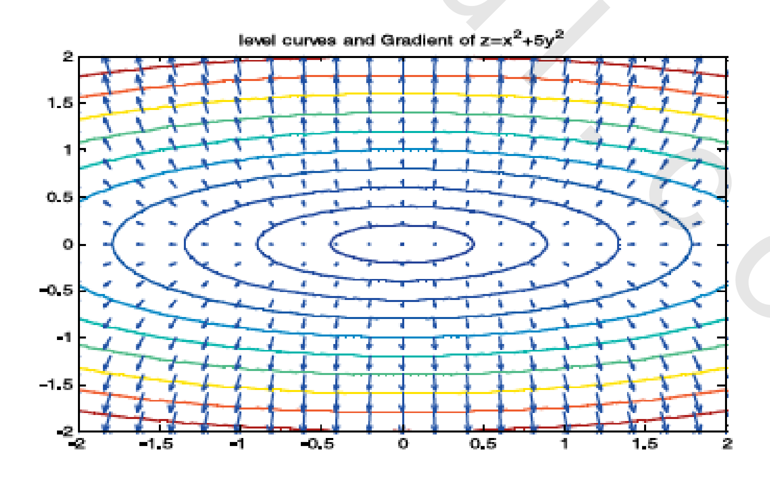
```
>> syms x y
>> f=sin(x^2+y^2);fx=diff(f,'x')
fx =
 2*cos(x^2+y^2)*x
 >> fxx=diff(f,'x',2)
fxx =
 -4*sin(x^2+y^2)*x^2+2*cos(x^2+y^2)
>> fxy=diff(fx,'y')
 fxy =
 -4*sin(x^2+y^2)*y*x
```

حقل المتجهات المعرّف على مجال الدالة f(x,y) يعرف بالتدرج Gradient $gradf(x,y) = \langle f_x, f_y \rangle$ أو $\nabla f(x,y)$ ، وهو عند كل نقطة يتعامد مع المنحنى السوي لـ f level curves عند تلك النقطة. MATLAB يساعد في رسم تدرج حقل المتجهات Gradient vector field و المنحنيات السوية للدالة.

مثال رقم (٤,١٨)

بالنسبة للدالة $f(x,y) = x^2+5y^2$ على $-2 \le x \ , y \le 2$ يمكن رسم المنحنيات السوية و رسم التدرج $\nabla f(x,y) = \langle 2x, 10y \rangle$ (الشكل رقم ٤,١٨) بالأوامر quiver و contour .

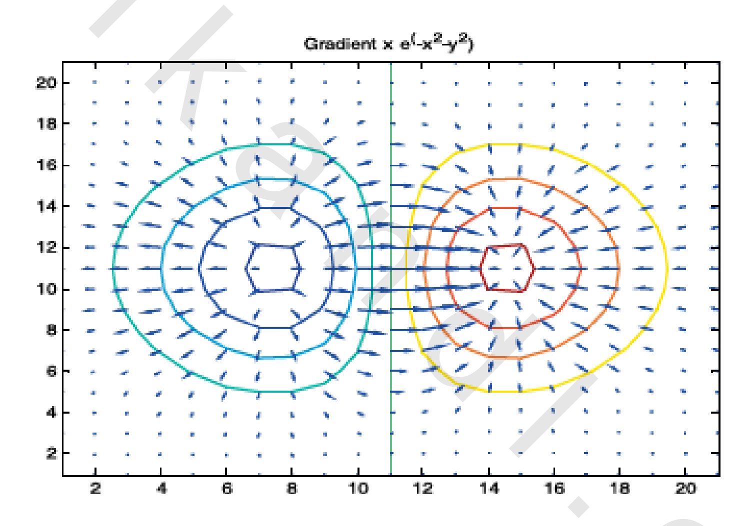
```
>> L=5*[0:.2:2].^2;
>> [x,y]=meshgrid(-2:.05:2);
>> [c,h]=contour(x,y,x.^2+5*y.^2,L);hold on
>> [x,y]=meshgrid(-2:.2:2);
>> dx=2*x;dy=10*y;
>> quiver(x,y,dx,dy)
```



الشكل رقم (٤,١٨). المنحنيات السوية و التدرج للدالة $z=x^2+5y^2$.

يمكن تقريب التدرج $\nabla f(x,y)$ بالأمر *gradient* . نقوم بتقدير التدرج للدالة $f(x,y) = xe^{-x^2-y^2}$ بعد تعريف مصفوفة المجال $-2 \le x \le 2$ و $-2 \le y \le 2$ باستعمال *meshgrid* ومن ثم نرسم (الشكل رقم ٤،١٩) باستخدام الأوامر quiver و contour :

```
>> [x,y] = meshgrid(-2:.2:2, -2:.2:2);
   z = x .* exp(-x.^2 - y.^2);
[px,py] = gradient(z,.2,.2);
 contour(z),hold on, quiver(px,py), hold off
```



الشكل رقم (٤،١٩). التدرج للدالة $xe^{-x^2-y^2}$.

(٤،٥،٧) القيمة العظمى والصغرى

رسم الدالة في متغيرين يساعد في تقدير القيمة العظمى والصغرى للدالة Maxima, Minima في مجال محدود ومغلق. الأمران max ، min يُستخدمان في حساب قيم ومكان هذه القيم بدقة على حسب حجم التقسيم المستخدم. فمثلاً الدالة

$f(x,y)=sin(x)+sin(y)+sin(x+y)$ على المنطقة $x \geq 0, y \leq \pi/4$ توجد لها قيمة عظمى عند $(\pi/4 , \pi/4)$ وقيمة صغرى عند $(-\pi/4 , -\pi/4)$.

نستخدم دالة *meshgrid* لتحديد مصفوفة المجال وتعريف الدالة :

```
>> [x,y]=meshgrid(-pi/4:pi/80:pi/4);
>> z=sin(x)+sin(y)+sin(x+y);
>> [m,ind]=max(z(1:prod(size(z))));
```

أما دالة *[m,ind] = max* فتساعد على تحديد القيمة العظمى وتخزنها في *m* وتخزن موقع القيمة العظمى في *ind* :

```
>> m
m =
   2.4142
>> num2str(x(ind))
```

ولتحويل الموقع *ind* إلى قيمة في مجال الدالة نستخدم *num2str* والناتج $(\pi/4 , \pi/4)$ حيث إن $\pi/4 = 0.7854$:

```
ans =
0.7854
>> num2str(y(ind))
ans =
0.7854
```

نكرر الخطوات لإيجاد القيمة الصغرى وذلك باستخدام *min* :

```
>> [mi,ind]=min(z(1:prod(size(z))));
>> mi
mi =
  -2.4142
>> num2str(y(ind))
ans =
-0.7854
>> num2str(x(ind))
ans =
-0.7854
```

(٤,٥,٨) تكامل متعدد Multiple Integral

الأمر *dbquad (f ,a,b,c,d)* يعطي قيمة عددية للتكامل الثنائي $\int_c^d \int_a^b f(x,y)\,dxdy$ حيث إن *f(x,y)* باستخدام الأمر *fcnchk* ودائما يبدأ *dbquad* بحساب التكامل بالنسبة لـ *x* من a إلى b ثم بالنسبة لـ *y* من c إلى d.

مثال رقم (٤,١٩)

احسب التكامل التالي :

$$\int_0^2 \int_0^1 \frac{1}{\sqrt{x^2+2y^2+1}}\,dxdy$$

```
>> in=dblquad(fcnchk('1./sqrt(x.^2+2*y.^2+1)'),0,1,0,2)
in =
   1.1597
```

(٤,٥,٨,١) دالة int الرمزية للتكامل المتعدد

طريقة أخرى لحساب التكامل الثنائي هي باستخدام دالة *int* في البيئة الرمزية *syms* على مرحلتين، الأولى بالنسبة لـ *x* والناتج يُجرى له تكاملاً بالنسبة للمتغير *y* لنحصل في النهاية على القيمة التقديرية للتكامل :

```
>> syms x y real
>> in=int(1./sqrt(x.^2+2*y.^2+1),x,0,1)
in =
asinh(1/(2*y^2+1)^(1/2))
>> inn=int(in,y,0,2)
inn =
 -2*log(3)+2*log(1+2^(1/2)*5^(1/2))-1/2*2^(1/2)*log(-2+5^(1/2))-
1/2*2^(1/2)*atan(2/5*5^(1/2))
>> double(inn)
ans =
   1.1597
```

مثال رقم (٤,٢٠)

لحساب التكامل الثلاثي للدالة $f(x,y,z)= z\,x^3y^2$ على الفترة
. $R=\{0\le y\le x\,,\,0\le z\le xy\,,\,0\le x\le 1\}$

$$\int_0^1\int_0^x\int_0^{xy} x^3y^2z\,dz\,dy\,dx$$

نكرر الأمر *int* ثلاث مرات ، مع تحديد متغير التكامل و حدود التكامل في كل جزء، لنحصل على النتيجة المطلوبة.

```
>> syms x y z real
>> ff=(x^3)*(y^2)*z

ff =

x^3*y^2*z
>> int(int(int(ff,z,0,x*y),y,0,x),0,1)

ans =
 1/110
```

(٤,٥,٨,٢) تطبيقات على التكامل المتعدد

التطبيق (١)

لحساب المساحة لسطح ما S (Surface area) حيث S هو المنحنى لدالة $z = f(x,y)$، نحتاج الى التحويل $r(x,y)=\langle x,y,f(x,y)\rangle$ ، و المساحة للسطح S تأخذ الصيغة .

$$\iint \sqrt{1+f_x^{\,2}+f_y^{\,2}}\,dxdy$$

ولتقدير المساحة السطحية لمجسّم القطع المكافئ في البعد الثالث Paraboloid معطى بالمعادلة $z = x^2+y^2$ ومعّرف على المجـــال $D = [0,1]x[0,1]$ نستعمل الأمر

dblquad والتكامل الذي يمثل المساحة السطحية هو:

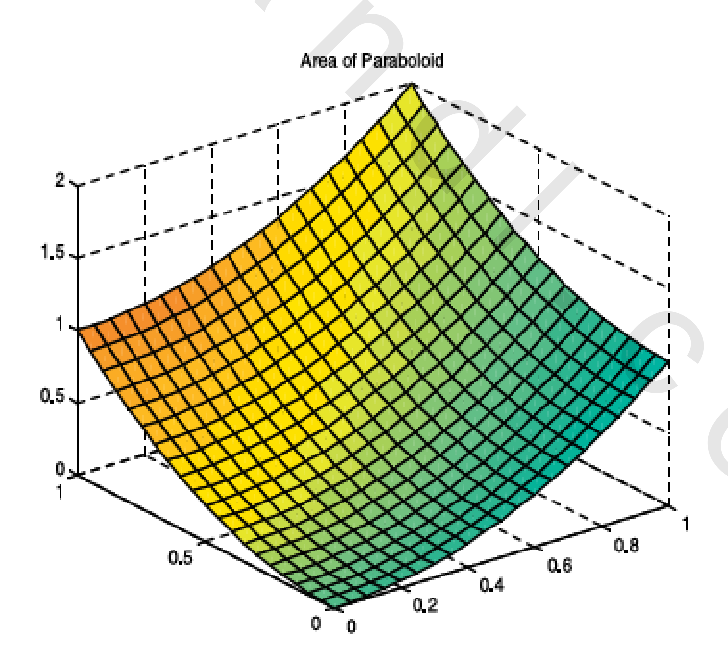
$$\int_0^1\int_0^1 \sqrt{1+4x^2+4y^2}\,dxdy \,.$$

ويمكننّا MATLAB من حساب مساحته بالأوامر التالية:

```
>> area=dblquad(fcnchk('sqrt(1+4*x.^2+4*y.^2)'),0,1,0,1);
>> area
area =
   1.8616
```

ونرسم السطح بالأوامر التالية (الشكل رقم ٤،٢٠):

```
>> [x,y]=meshgrid(0:.05:1);
>> surfl(x,y,x.^2+y.^2);
```



الشكل رقم (٤،٢٠). المساحة السطحية لمجسّم القطع المكافئ.

التطبيق (٢)

احسب مساحة المنطقة المحصورة بين المنحنيين القطبيين :

$$r = 6$$
$$r = 3\sec\theta \quad for \quad x \geq 3$$

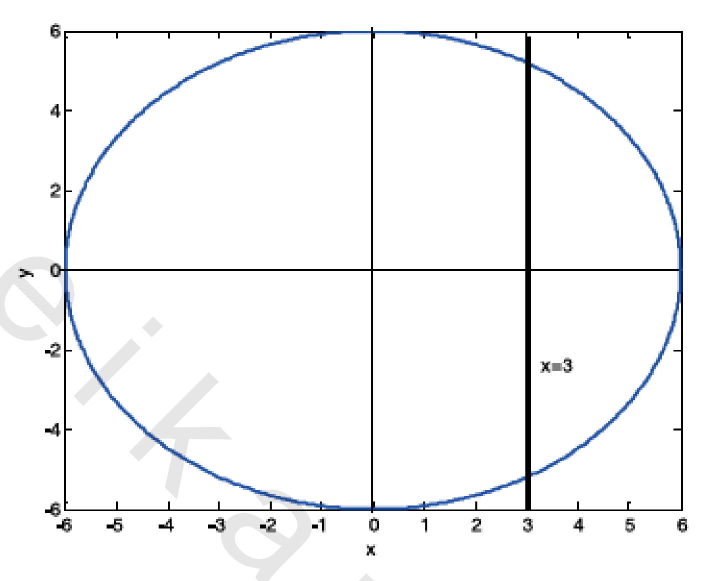
حيث إن $r = 6$ معادلة دائرة، وقيمة نصف القطر هي 6 ومركزها نقطة الأصل. أما $r = 3\sec\theta$ فهي معادلة خط مستقيم $x=3$. نحتاج نقطة التقاطع ونجدها بدالة *solve* لحل المعادلة $3\sec\theta - 6 = 0$.

```
>> syms t r real
>> solve(3*sec(t)-6)
ans =
1/3*pi
```

ثم المساحة المطلوبة تقدر بالتكامل $\int_{-\pi/3}^{\pi/3} d\theta \int_{3\sec\theta}^{6} r dr$ و نستخدم الأمر *int* في *syms* للحصول على التكامل :

```
>> c=int(r,3*sec(t),6)
 c =
 18-9/2*sec(t)^2
>> int(c,-pi/3,pi/3)
ans =
12*pi-9*3^(1/2)
```

الشكل رقم (٤،٢١). يوضح المنطقة المطلوب إيجاد تكاملها وهي المساحة المحصورة بين المنحنيين.



الشكل رقم (٤,٢١). المساحة المحصورة بين المنحنيين.

(٤,٦) تماريـــــن

١- قدر المجموع $\sum_{1}^{\infty} i^{-4}$.

٢- احسب أ) $\lim_{x \to 0+} x^{\sqrt{x}}$

ب) $\lim_{x\to\infty+} x\sin(\frac{1}{x})$

٣- استخدم برنامج *simpsons* لحساب التكاملين على ٣٢ فترة، مع العلم أن القيمة الحقيقية للتكامل هي C(1)=0.779834 و S(1)=0.4382591 .

$$S(1)= \int_0^1 \cos(\pi t^2/2)\,dt$$

$$C(1)= \int_0^1 \sin(\pi t^2/2)\,dt$$

٤- عدل برنامج سمبسون المركب ليقدر تكاملاً متعدداً ومن ثم احسب التكامل التالي باستخدام 64 تجزئة .

$$\int_{-1}^{1} dy \int_{-\pi}^{\pi} x^4 y^4\, dx$$

٥- استخدم دالة *diffgen* لإيجاد المشتقة الأولى والثانية للدالة $x^2\cos x$ عندما $x = 1$ استخدم $h= 0.1$ ثم عند $h = 0.01$.

٦- أوجد المساحة المحصورة بين منحنى الدالتين $y = \exp(x), y = \ln(x)$ بين $x = 2$ و $x = 5$.

٧- احسب حجم الجسم الناتج عن دوران $y = x^2-6$ حول محور y عند $y = 0$ و $y = 10$.

٨- ارسم الدالة $z = \sin(x^2+y^2)$ والمستوى المماس عند النقطة (1, 1, sin2) .

٩- قرب التكامل :

أ) $\int_0^1 \frac{e^x}{\sqrt{x}}\,dx$

ب) $\int_1^\infty x^{-3/2}\sin(1/x)\,dx$

جـ) $\int_{-\infty}^{\infty} \frac{1}{1+x^2}\,dx$

د) $\int_0^\infty \frac{1}{1+x^4}\,dx$

هـ) $\int_0^\infty \frac{1}{(1+x^2)^3}\,dx$

و) $\int_0^\infty e^{-x}\sin x\,dx$

الفصل الخامس

حلول المعادلات التفاضلية على MATLAB

المعادلات التفاضلية هي معادلات ذات أهمية خاصة لدى المهندسين، والفيزيائيين والرياضيين، لأن ظواهر عديدة فيزيائية، بيولوجية، وكيميائية، واقتصادية تُمثل رياضياً بهذه المعادلات. في هذا الفصل نوضح كيفية استخدام MATLAB لإيجاد حلول عددية لبعض أنواع المعادلات التفاضلية.

(٥,١) مقدمة في المعادلات التفاضلية

المعادلة التفاضلية العادية Ordinary differential equation هي معادلة تتضمن دالة $y = f(t)$ في متغير t ومشتقاتها $y', y'', \ldots, y^{(n)}$ على شكل:

$$F(t, y, y', y'', \ldots, y^{(n)}) = 0$$

وتعرّف رتبة المعادلة التفاضلية order بأنها رتبة المشتق ذي الرتبة العليا الذي يظهر فيها. تحسب درجة المعادلة degree بأخذ درجة الحد الأعلى رتبة. وتسمى المعادلة التفاضلية خطية linear إذا كانت y وجميع مشتقاتها من الدرجة الأولى ومعامل y

وجميع معاملات مشتقاتها بمتغير واحد t أو ثابت، وإذا كانت المعادلات التفاضلية تحتوي على دالة في متغيرين أو أكثر وتفاضل جزئي، فتسمى معادلات تفاضلية جزئية partial differential equation. نطلق على الدالة التي تحقق المعادلة التفاضلية حل المعادلة التفاضلية solution، فمثلاً :

$$y'+y''=0$$

هي معادلة تفاضلية عادية خطية من الرتبة الثانية والدرجة الأولى، أما الحل فهو دالة على شكل $y=e^{-t}$.

هناك معادلات تفاضلية بقيمة ابتدائية معطاة $y(t_0) = y_0$ وتسمى معادلات تفاضلية بقيم ابتدائية Initial value problem وأخرى بقيم عند حدود المجال، وتسمى Boundary value problems. كما أن هناك طرقاً تحليلية للتوصل للحل الفعلي للمعادلات التفاضلية، ولكن بعضاً من المعادلات لا يوجد لها حل فعلي، أو أن الحل الفعلي يصعب التعامل معه، لذلك نلجأ إلى طرق عددية للحصول على حل للمعادلة. وهذا يعني أننا نقدر الحل المتصل للمعادلة بقيم تقريبية متقطعة تعطي قيمة دالة الحل عند نقاط معينة في المجال تبدأ بنقطة ابتدائية وتنتهي عند نقطة نهائية معطاة. فنقرب $y(t_i)$ بالقيمة y_i للقيم t_i بحيث إن $t_i = t_0 + ih$ ، h طول الفترة و $i= 0,1,2,...n$.

في هذا الفصل سنوضح كيفية استخدام MATLAB في حل معادلات تفاضلية بطرق عددية ، كما سنعطي دوال جاهزة في MATLAB تحسب الحلول للمعادلات التفاضلية في البيئة المعتادة أو في البيئة الرمزية syms .

(٥,٢) طريقة أويلر Euler Method

لحل معادلة تفاضلية بقيمة ابتدائية توجد طرق ذات خطوة واحدة وأخرى بعدة

خطوات. تُعد طريقة أويلر *Euler* من طرق الخطوة الواحدة، ولا تحتاج لقيمة ابتدائية سوى المعطاة في المعادلة التفاضلية. و هي من أبسط الطرق العددية المباشرة لحل معادلة تفاضلية عادية من الرتبة الأولى، مثل :

$$y' = \frac{dy}{dt} = f(t,y) \qquad a \le t \le b, \qquad y(a) = y_0$$

لاستنتاج صيغة أويلر نستخدم سلسلة تايلور على دالة الحل $y(t)$:

$$y(t_i + h) = y(t_i) + y'(t_i)h + y''(\mu)h^2/2$$

لتأخذ طريقة أويلر الصيغة :

$$y(t_{i+1}) = y(t_i) + hy'(t_i)$$

حيث إن $h = t_{i+1} - t_i$ و العدد μ بين (t_i, t_{i+1}) والخطأ برتبة h^2 و يقدر بـ $(h^2/2)y''(\mu)$.

لذا يمكن كتابة صيغة أويلر النهائية:

$$y_0 = y(a)$$
$$y_{i+1} = y_i + hf(t_i, y_i)$$

لكل $i=0,1,2,3,\dots n-1$ و $y_i=y(t_i)$. وتنفذ طريقة Euler بالخوارزمية (٥,١):

```
function [tvals, yvals]=feuler(f,start,finish,startval,h)
s=(finish-start)/h+1;
y=startval;t=start;
yvals=startval;tvals=start;
for i=2:s
  y1=y+h*feval(f,t,y);
 t1=t+h;
  tvals=[tvals, t1];
yvals=[yvals, y1];
  t=t1;
  y=y1;
end
```

خوارزمية (٥,١)

مثال رقم (٥,١)

استخدم الخوارزمية (٥,١) لحل المعادلة التفاضلية $y' = xy + x$ بقيمة ابتدائية $y(0) = 0$ عند $0 \leq x \leq 1$ و $h=0.1$.

الحل :

بعد تخزين الدالة في m-file يدعى *fun* ندخل :

```
[x,y]=feuler('fun',0,1,0, 0.1);
>> plot(y,'r');
>> hold;
>> plot(-1+exp((x.^2)/2));
```

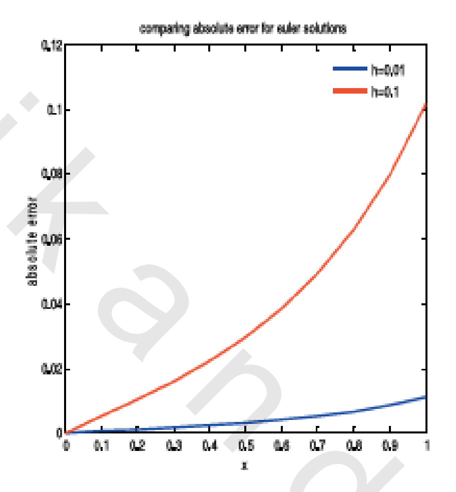
تظهر نتائج المتغيرات x , y ويمكن مقارنتها بالحل الفعلي $y(x) = e^{x^2/2} - 1$ عند نفس النقاط في الجدول رقم (٥,١).

الجدول رقم (١). نتائج مثال رقم (٥,١).

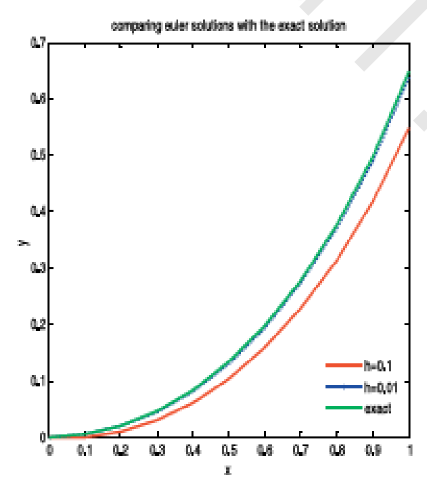
x	y	exact
0.1000	0	0.0050
0.2000	0.0100	0.0202
0.1000	0	0.0050
0.2000	0.0100	0.0202
0.3000	0.0302	0.0460
0.4000	0.0611	0.0833
0.5000	0.1036	0.1331
0.6000	0.1587	0.1972
0.7000	0.2283	0.2776
0.8000	0.3142	0.3771
1.0000	0.5471	0.6487

من الرسم نقارن الحل العددي الناتج من طريقة أويلر بالحل الفعلي ، ويمكن تحسين الحل بتصغير الخطوة h إلى $h= 0.01$ كما يظهر في الشكل رقم (٥,٢)، كما

يمكننا مقارنة الخطأ المطلق في الشكل رقم (٥,١) لنفس قيم *h*. ورغم أن طريقة أويلر يندر استخدامها عملياً(لكبر قيمة الخطأ) ، فإنه يمكن الاستفادة من بساطة استنتاجها لتوضيح الأساليب التي يتضمنها إنشاء بعض الطرائق الأكثر تقدماً.



الشكل رقم (٥,١). مقارنة الخطأ المطلق لحل أويلر التقريبي للمعادلة.



الشكل رقم (٥,٢). مقارنة الحل الفعلي وحل أويلر التقريبي للمعادلة.

(٥،٣) طريقة رونج كوتا Runge Kutta Method

تتكون طريقة رونج كوتا Runge-Kutta من مجموعة من الطرق العددية لحل المعادلات التفاضلية وهي عملية جداً وتعطي حلولاً بدقة عالية. ولطرائق رونج كوتا خطأ قطع محلي local truncation error ذو رتبة عالية ، بينما لا تحتاج إلى حساب وتقدير مشتقات *f(x,y)* مما يوجد في الطرق الأخرى.

هناك طرق لرونج كوتا من الدرجة الثانية second order Runge-Kutta methods مثل طريقة هيون Heun method وطريقة النقطة الوسيطة Midpoint method وطريقة أويلر المعدلة Modified Euler method . وتُعد هذه العائلة من طرق رونج كوتا سهلة البرمجة ، وتستخدم أحياناً لإيجاد نقاط ابتدائية لبرامج أخرى. فمثلاً بفرض $y_0 = y(a)$ ولكل $i=0,1,2,3,...n-1$

طريقة هيون تأخذ الصيغة العامة :

$$y_{i+1} = y_i + \frac{h}{2}(k_1 + k_2)$$
$$k_1 = f(t_i, y_i) \quad , \mathrm{k}_2 = f(t_{i+1}, y_i + hk_1)$$

و صيغة طريقة النقطة الوسيطة:

$$y_{i+1} = y_i + hf(t_i + \frac{h}{2}, y_i + \frac{h}{2} f(t_i, y_i))$$

وطريقة أويلر المعدلة تأخذ الصيغة :

$$y_{i+1} = y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, y_i + hf(t_i, y_i))]$$

كما توجد طرق رونج كوتا من درجات أعلى مثل رونج كوتا الرابعة

الكلاسكية Classical RK4 ذات الخطأ برتبة $O(h^4)$:

Runge Kutta 4

$$k_1 = hf(t_i, y_i)$$
$$k_2 = hf(t_i + \frac{h}{2}, y_i + \frac{k_1}{2})$$
$$k_3 = hf(t_i + \frac{h}{2}, y_i + \frac{k_2}{2})$$
$$k_4 = hf(t_i + h, y_i + k_3)$$
$$y_{i+1} = y_i + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

أو رونـــج كوتا ميرســـونRK Merson ذات الخطأ برتبة $O(h^5)$ [28] و تأخذ الصيغ التالية بفرض $y_0 = y(a)$ ولكل $i=0,1,2,3,...n-1$:

Runge Kutta Merson

$$k_1 = hf(t_i, y_i)$$
$$k_2 = hf(t_i + \frac{h}{3}, y_i + \frac{k_1}{3})$$
$$k_3 = hf(t_i + \frac{h}{3}, y_i + \frac{k_2}{6} + \frac{k_1}{6})$$
$$k_4 = hf(t_i + \frac{h}{2}, y_i + \frac{k_1}{8} + \frac{3k_3}{8})$$
$$k_5 = hf(t_i + h, y_i + \frac{k_1}{2} - \frac{3k_3}{2} + 2k_4)$$
$$y_{i+1} = y_i + \frac{1}{6}[k_1 + 4k_4 + k_5]$$

يمكن تطبيق الخوارزمية rgen (الملحق) [15] التي تسمح للمستخدم الاختيار ما بين الطريقتين المختلفتين من رونج كوتا عند مناداة البرنامج .

مثال رقم (٥,٢)

قارن حل المعادلة $y' = 2xy$ إذا كان الشرط الابتدائي $y(0)=2$ و x من 0 إلى 2 باستخدام طريقة Kutta– Rung الكلاسكية و ميرسون ، حيث إن $h=0.2$ والحل الحقيقي $y = 2e^{x^2}$.

الحل :

الأوامر التالية تُنفذ على ملف الدالة f501 :

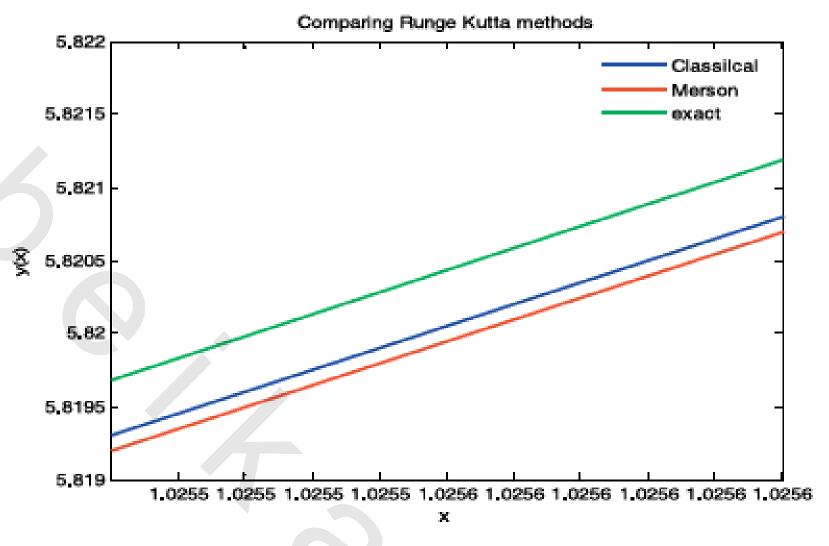
```
function yp=f501(x,y)
   yp=2*x*y;
```

ثم نكرر استخدام الخوارزمية باختيار الرقم 1 لرونج كوتا الرابعة الكلاسكية Classical RK4 أو الرقم 2 لرونج كوتا ميرسون RK Merson ونرسم النتائج الموضحة في الشكل رقم (٥,٣):

```
[tvals,yvals1]=rkgen('f501',0,2,2,0.2,1);
[tvals,yvals2]=rkgen('f501',0,2,2,0.2,2);
y=2.*exp(tvals.^2);
```

```
plot(tvals,yvals1)
hold
Current plot held
plot(tvals,yvals2,'r')
plot(tvals,y,'g')
```

الطرق تعطي نتائج متقاربة جداً، ولذلك قمنا بالتقريب باستخدام zoom في شريط مهام الشكل، عندها لاحظنا أن الحل بطريقة رونج كوتا الرابعة الكلاسكية أدق مقارنة بطرق رونج كوتا ميرسون، ولكن من عيوب كل طرق رونج كوتا كمية الحسابات الكثيرة وعدد مرات التعويض بالدالة f .



الشكل رقم (٥،٣). مقارنة طرق رونج كوتا.

(٥،٤) طريقة التخمين والتصحيح Predictor- Corrector Methods

تدعى الطرائق السابقة طرائق الخطوة الواحدة لأن التقريب يتم عن طريق نقطة واحدة سابقة، ولكن عملياً نحتاج إلى استخدام أكثر من نقطة. والطرق السابقة ضرورية للحصول على القيم الأولية، إلا أنها عموماً تتطلب جهداً للحصول على الحل العددي بالدقة المطلوبة. أما الطرائق ذات الخطوات المتعددة فهي نوعان، منها الطرائق الضمنية implicit methods والطرائق الصريحة explicit methods. نعرض هنا طرقاً تطلق عليها طرق التخمين والتصحيح Predictor-Corrector وهي توليف بين طريقة صريحة وأخرى ضمنية، حيث تخمن الطريقة الصريحة التقريب، وتصحح الطريقة الضمنية هذا التخمين. هناك طرق عديدة متعددة الخطوات تستخدم للتخمين والتصحيح نعرض إحداها وهي:

١- للتخمين: طـــريقة آدمز- باشفورث Adams Bashforth ذات الخطوات الأربع و خطأ برتبة $O(h^4)$: و $i=3,4,5,...n-1$.

$$y_0=\alpha_0, \quad y_1=\alpha_1, \quad y_2=\alpha_2, \quad y_3=\alpha_3,$$
$$y_{i+1}=y_i+\frac{h}{24}(55f(t_i,y_i)-59f(t_{i-1},y_{i-1})+37f(t_{i-2},y_{i-2})-9f(t_{i-3},y_{i-3}))$$

٢- للتصحيح : طريقة آدمز- مولتون Adams-Moulton method ذات الخطوات الثلاث و خطأ برتبة $O(h^4)$ و $i=3,4,5,...n-1$.

$$y_0=\alpha_0, \quad y_1=\alpha_1, \quad y_2=\alpha_2,$$
$$y_{i+1}=y_i+\frac{h}{24}(9f(t_{i+1},y_{i+1})+19f(t_i,y_i)-5f(t_{i-1},y_{i-1})+f(t_{i-2},y_{i-2}))$$

وفي المثـال (٥،٣) التـالي نـستخدم Adams- Bashforth لإيجـاد الحـل التقـريبي للمعادلة التفاضلية ، ثم نحسن الحل بطريقة Adams-Moulton method.

مثال رقم (٥،٣)

حـل المعادلـة $y'=-5y$ عنـدما $y=50$ و في الفـترة $t=0$ إلى $t=6$ و h تأخـذ القيــم 0.2 ، 0.1 و 0.4 .

الحل :

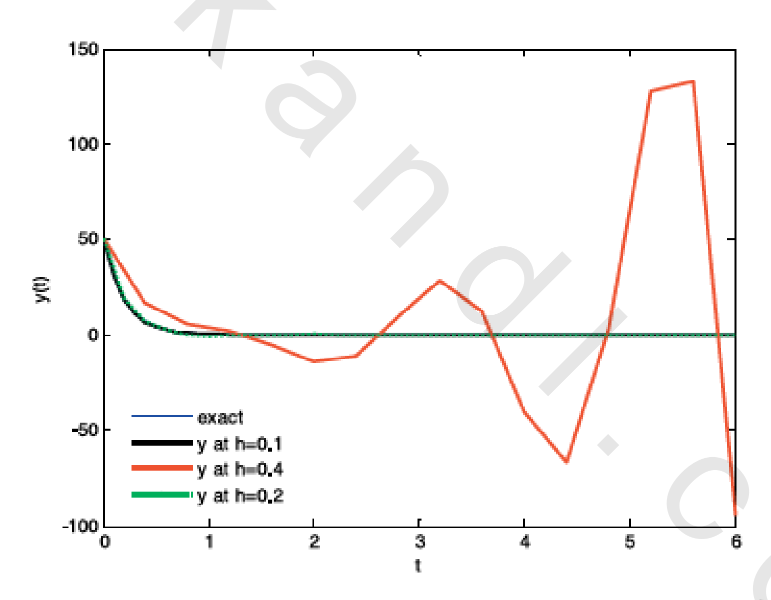
نكتب الدالة في m-file ونطبق البرنامج abm (الملحق) [15] ، مع العلم أن الحل الفعلي $y=50e^{-5t}$.

```
function yp=f5(t,y)
yp=-5*y;
```

ثم نطبق البرنامج :

```
[t1 y1]=abm('f5',0,6,50,0.1);
[t2 y2]=abm('f5',0,6,50,0.2);
[t3 y3]=abm('f5',0,6,50,0.4);
```

من الشكل رقم (٥,٤) نلاحظ أن الحل مستقرٌ عندما h تساوي 0.1 و 0.2 ، وغير مستقر عند 0.4.



الشكل رقم (٥,٤). رسم الحلول التقريبية بخطوات مختلفة بطريقة predictor- corrector .

(٥,٥) دوال MATLAB لحل المعادلات التفاضلية

توجد دوال جاهزة على MATLAB تقوم بحل المعادلات التفاضلية وهي : *ode45, ode15s, ode113, ode23, ode23s* و *ode23t*. نقدم شرحاً مبسطاً لبعضها في الجدول رقم

(٥,٢) مع الإشارة إلى أن اختيار الدالة المناسبة يعود لنوع المعادلة و دقة الحل المطلوب.

الجدول رقم (٢). دوال جاهزة لحل المعادلات التفاضلية.

الدالة	الوظيفة
Ode45	وهي عبارة عن نظام حل بخطوة واحدة، أي أنها تحتاج عند حساب الحل في اللحظة t_n لمعرفة القيمة التي تسبقها مباشرة t_{n-1} ويستخدم طريقة رونج كوتا ٤ او٥ الصريحة. وتأخذ الصيغة العامة الشكل [t,y] = ode45(fun,tspan,y0) ، يجب أن تكون fun على شكل الطرف الأيمن من $y' = f(x,t)$ أو $M(t,y)y' = f(x,t)$ و tspan هو مجال تغير t أما y0 فهو شرط البداية للمعادلة. وفي الأغلب تجرب كأول طريقة لحل المسائل غير الجامدة.
Ode23	طريقة أكثر فعالية وأدق من سابقتها، وهي أيضاً طريقة وحيدة الخطوة ، وتستخدم رونج كوتا ٢ أو ٣ الصريحة . الصيغة العامة : [t,y] = ode23(fun,tspan,y0).
Ode113	تستخدم طريقة الخطوات المتعددة التخمين والتصحيح ادمز- باشفولد- مولتون Adams-Bashfold-Moulton predictor corrector method.
Ode23s	تفضل هذه الدالة في حالة المسائل العنيدة و تستخدم طريقة روزنبروك المحسنة Modified Rosenbrock.
dsolve	الطريقة المتبعة لإيجاد حل معادلة تفاضلية عادية بشرط ابتدائي في البيئة الرمزية الصيغة العامة ويرمز لمعامل التفاضل بـ D dsolve('eqn1','x(t)=a',...) .

مثال رقم (٥,٤)

إذا كانت المعادلة التفاضلية :

$$2\frac{d^2x}{dt^2}+4\left(\frac{dx}{dt}\right)^2 - 2x = \cos x$$

والشرط الابتدائي $x=0$ و $dx/dt=10$ عندما $t=0$.أوجد حل المعادلة التفاضلية باستخدام ode23 وode45 في الفترة من 1 إلى 2.

الحل :

نعرف الدالة في ملف خاص m-file :

```
function fv=f54(t,x)
global p
fv=zeros(2,1);
fv(1)=x(1);
fv(2)=0.5*cos(x(2))+x(2)-(2*(x(1)^2));
```

وللوصول للحل نطبق الأوامر التالية :

```
[t1 ,x1]=ode23(@f54,[1,2],[0,10]);
[t2 , x2] = ode45(@f54,[1,2],[0,10]);
```

(٥,٥,١) دالة dsolve الرمزية

الأمر *dsolve* يقدم حلولاً لمعادلات تفاضلية عادية في بيئة syms. وتكتب الدالة بصيغة رمزية ، ويمكن أيضاً إدخال نظام من المعادلات التفاضلية.وتأخذ الصيغة العامة :

dsolve('eqn1','eqn2', ...)

يرمز في كتابة المعادلة التفاضلية على syms لمعامل التفاضل 'D' وهو بالنسبة

للمتغير المستقل 't' . إذا تلا D حرف فهو المتغير غير المستقل، أما إذا تلاه رقم فهو تفاضل متكرر، فالتفاضل الثاني للمتغير y(t) هو D2 y .

القيم الابتدائية تكتب على شكل معادلات مثل 'y(a)=b' أو 'Dy(a) = b' . وإذا كان عدد القيم الابتدائية أقل من عدد المتغيرات المستقلة، فإن الحل سيحتوي على ثوابت C1, C2 . أما إذا كان الحل ليس صريحاً فإن الحل سيكون ضمنياً وستظهر جملة تنبيه لذلك. و إذا كان لم يكن هناك حل ضمني ولا حل صريح، فتعطى جملة تنبيه وحل فارغ يتضمن *كلمة sym* . في حال كانت المعادلات التفاضلية غير خطية، فالحل سيتضمن معادلة تفاضلية من درجة منخفضة.

مثال رقم (٥,٥)

أوجد حل المعادلة التفاضلية في البيئة الرمزية :

$$y'^2 + y^2 = 1 \qquad y(0) = 0$$

الحل :

نقوم بإدخال الأوامر التالية :

```
y = dsolve('(Dy)^2 + y^2 = 1','y(0) = 0')
y=
[sin(t ) ]
[Sin(-t)]
```

وهنا نعرض أمثلة أخرى للحالات المختلفة الأخرى :

```
>>Y = dsolve('Dy = y^2*(1-y)')
        Warning: Explicit solution could not be found; implicit solution
returned.
        Y =
        t+1/y-log(y)+log(-1+y)+C1=0
```

تنبيه بوجود حل ضمني

```
>> dsolve('Df = f + sin(t)', 'f(pi/2) = 0')
     ans =
     -1/2*cos(t)-1/2*sin(t)+1/2*exp(t)/(cosh(pi)+sinh(pi))^(1/2)
```

```
>> dsolve('D2y = -a^2*y', 'y(0) = 1, Dy(pi/a) = 0')
        ans =
        cos(a*t)
```

ويمكن حل معادلة أخرى :

```
>>syms s x p t
s=dsolve('D2x+Dx+.25*x=0','Dp+.5*p+.25*x=0','x(0)=4','Dx(0)=0','p(0)=0')
s =
  p: [1x1 sym]
  x: [1x1 sym]
>> s.x
 ans =
 exp(-1/2*t)*(4+2*t)
 >> s.p
 ans =
 -1/8*(8*t+2*t^2)*exp(-1/2*t)
```

مثال رقم (٦, ٥)

أوجد حلاً للمعادلة التفاضلية :

$$\frac{d^3u}{dx^3} = u \qquad \text{with} \quad u(0) = 1, \quad u'(0) = -1, \quad u''(0) = \pi$$

الحل :

```
>> dsolve('D3u=u','u(0)=1','Du(0)=-1','D2u(0)=pi','x')
 ans =
 1/3*pi*exp(x)-1/3*(1+pi)*3^(1/2)*exp(-1/2*x)*sin(1/2*3^(1/2)*x)+(1-
1/3*pi)*exp(-1/2*x)*cos(1/2*3^(1/2)*x)
```

(٥,٦) معادلات تفاضلية جزئية على MATLAB

سنتناول في هذا الجزء الحل العددي ببرامج على MATLAB لمسائل تحوي معادلة تفاضلية بمشتقات جزئية تتعلق بمسائل فيزيائية مختلفة ، وسنقتصر على المعادلات الكلاسيكية مثل المعادلات التفاضلية الجزئية الناقصية ، والتكافئية والزائدية :

$$a\frac{\partial^2 u}{\partial x^2}+b\frac{\partial^2 u}{\partial x\partial y}+c\frac{\partial^2 u}{\partial y^2}+d\frac{\partial u}{\partial x}+e\frac{\partial u}{\partial y}+fu+g=0$$

حيث إن a,b,c,d,e,f,g إما مقادير ثابتة وإما دوال في المتغيرين x و y.

تصنف المعادلة بأنها تكافئية عندما :

$$b^2-4ac=0$$

ونقول إنها ناقصية عندما :

$$b^2-4ac<0$$

وإنها معادلة زائدية عندما :

$$b^2-4ac>0$$

(٥,٦,١) المعادلات التفاضلية الجزئية الناقصية

Elliptic Partial-Differential Equations

سنقوم بدراسة المعادلة التفاضلية الجزئية الناقصية المعروفة باسم معادلة بواسون Poisson equation وهي :

$$\nabla^2 u(x,y) \equiv \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y)$$

لكل $(x,y) \in R$ وحيث إن:

$$(x,y) \in S, \forall u(x,y) = g(x,y)$$
$$R = \{(x,y) | a < x < b, c < y < d\}$$

ونرمز لحدود R بـ S مع الفرض هنا أن كلاً من f و g متصلة على مجالها، وهذا يضمن وحدانية الحل.

الطريقة المستخدمة هي تعديل لطريقة الفرق – المحدود لمسائل القيمة الحدية finite–difference method .

الخطوة الأولى هي اختيار عددين n و m وتعريف مقياس الخطوتين h و k بـ $h = \frac{(b-a)}{n}$ و $k = \frac{(d-c)}{m}$ ثم تُجزأ الفترة $[a,b]$ إلى n من الأجزاء المتساوية بطول h والفترة $[c,d]$ إلى m من الأجزاء المتساوية بطول k. باستخدام متسلسلة تايلورنصل للصيغ التالية:

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j),$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{k^2} - \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \mu_j)$$

حيث إن: $\xi_i \in (x_{i-1}, x_{i+1})$, $\mu_j \in (y_{j-1}, y_{j+1})$.

نستخدم القانونين في معادلة بواسون وذلك بالتعويض في المشتقات الجزئية بالصورة التالية :

$$\frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} + \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{k^2}$$
$$= f(x_i, y_j) + \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \mu_j)$$

لكل $i = 1,2,......, n-1$ ولكل $j = 1,2,.....,m-1$ والتعبير عن الشروط الحدية :

$u(x_0 , y_j) = g(x_0 , y_j)$ لكل $j = 0,1......m$

$u(x_n , y_j) = g(x_n , y_j)$ لكل $j = 0,1......m$

$u(x_i , y_0) = g(x_i , y_0)$ لكل $i = 0,1......n-1$

$u(x_i , y_m) = g(x_i , y_m)$ لكل $i = 0,1......n-1$

باستخدام طريقة الفرق– المحدود واستبدال الحدود التفاضلية بالقيم التقريبية الناتجة من متسلسلة تايلور نحصل على معادلة الفــرق حيث إن $x_i=a+ih, y_j=c+jk$ ذات خطأ قطع برتبة $O(k^2+h^2)$:

$$2[(h/k)^2+1]u_{i,j}-(u_{i+1,j}+u_{i-1,j})-(\frac{h}{k})^2(u_{i,j-1}+u_{i,j+1})=-h^2 f(x_i,y_j)$$

وبذلك تُحول المعادلة التفاضلية إلى نظام من المعادلات الخطية ، والذي يعد حله هو الحل العددي للمعادلة التفاضلية الجزئية الأصلية. نستخدم برنامج MATLAB لحل هذا النظام حسب الطرق الواردة في الفصل الثالث. نطبق خوارزمية معادلة بواسون للفرق المحدود (الملحق) في المثال رقم (٥،٥).

مثال رقم (٥،٧)

استخدم MATLAB لتقريب حل المعادلة التفاضلية الناقصية ، وقارن النتائج مع الحل الفعلي $u(x,y)=(x-y)^2$:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=4, \quad 0<x<1, \quad 0<y<2;$$
$$u(x,0)=x^2, \quad u(x,2)=(x-2)^2, \quad 0\le x\le 1;$$
$$u(0,y)=y^2, \quad u(1,y)=(y-1)^2, \quad 0\le y\le 2;$$

الحل :

نطبق البرنامج *poison* (الملحق) التي تطلب إدخال البيانات الضرورية حسب التالي :

```
>> poison

The Elliptic equation of the form
d^2u/dt^2 + d^2u/dx^2 = f(x,y)  a<x<b,  c<y<d
Subject to the boundary conditions

u(a,y)=g(a,y)       c<y<d
```

```
u(b,y)=g(b,y)
u(x,c)=g(x,c)        a<x<b
u(x,d)=g(x,d)
```

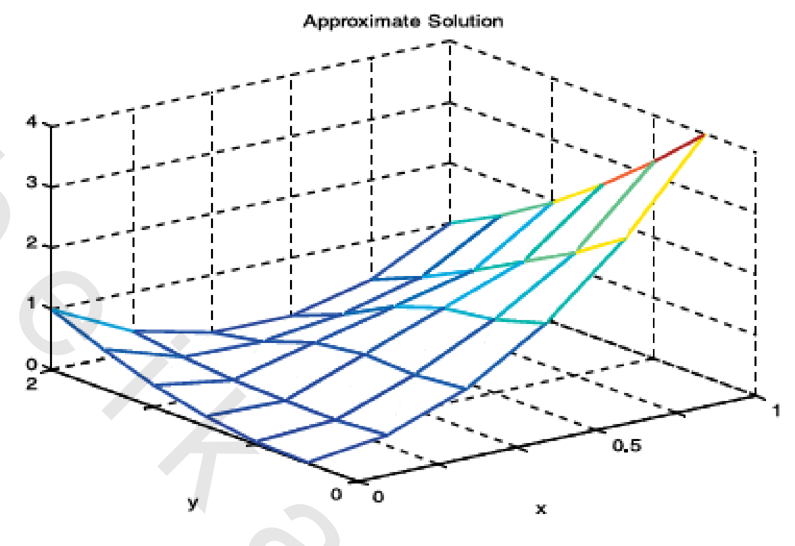
```
the number of grid sections for the x variable; n=  5
enter the number of grid sections for the y variable; m =  5
enter the maximum number of iterations  50
enter the right end point of the range of  x ; a=  0
enter the left end point of the range of  x ; b=  1
enter the right end point of the range of  y ; c=  0
enter the left end point the range of  y ; d=  2
enter the tolerance  ; tol=.0005
enter the function f(x,y)=4
enter the exact solution e(x,y)=(x-y).^2
enter boundary condition  u(x,c)=(x).^2
enter boundary condition  u(x,d)=(x-2).^2
enter boundary condition  u(a,y)=(y).^2
enter boundary condition  g(b,y)=(1-y).^2
```

يظهر لنا الحل الفعلي و التقريبي (الشكل رقم ٥,٥) ومربع الخطأ التام total squared error :

```
The exact solution is  e =
 0.1600    0.6400    1.4400    2.5600    4.0000         0
   0.0400    0.0400    0.3600    1.0000    1.9600    3.2400
 0.1600    0.6400    1.4400    2.5600       0.1600        0
   0.3600    0.0400    0.0400    0.3600    1.0000    1.9600
0.1600    0.6400    1.4400       0.6400    0.1600        0
   1.0000    0.3600    0.0400    0.0400    0.3600    1.0000

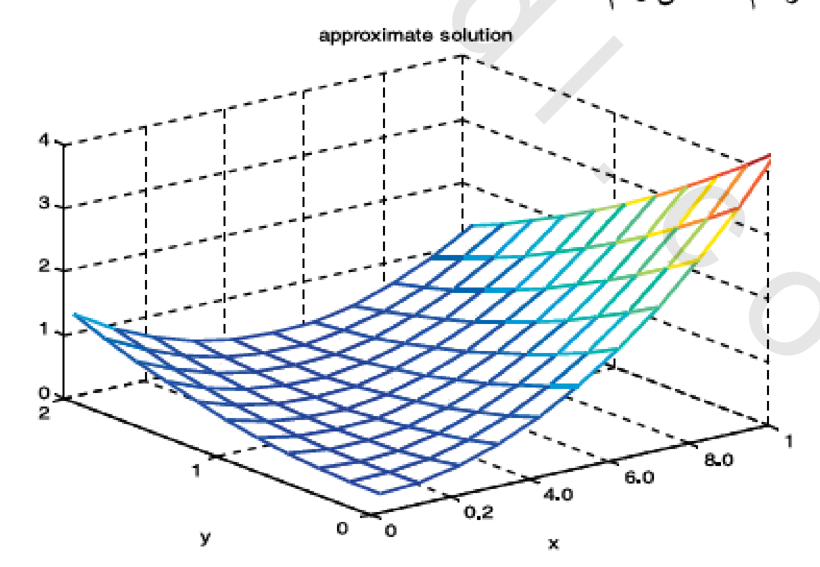
The approximated solution is  w =
0.1600    0.6400    1.4400    2.5600    4.0000         0
   0.0400    0.1081    0.5644    1.2043    2.0279    3.2400
   0.1600    0.1192    0.6032    1.0830    1.5589    2.5600
   0.3600    0.1592    0.4833    0.8032    1.1190    1.9600
   0.6400    0.2282    0.2046    0.3645    0.7080    1.4400
   1.0000    0.3600    0.0400    0.0400    0.3600    1.0000

error =    1.0280
```



الشكل رقم (٥,٥). الحل الناتج لمعادلة بواسون m=n=5 .

كلما كبرت قيمة m و n فإن الحل يتحسن ، والرسم يصبح أدق وهذا ما يتضح من الرسم (الشكل رقم ٥,٦) عند m=n=10.



الشكل رقم (٥,٦). الحل الناتج لمعادلة بواسون بقيم m=n=10 .

(٥,٦,٢) المعادلات التفاضلية الجزئية التكافئية

Parabolic Partial-Differential Equations

نعرض معادلة الحرارة أو الانتشار، وهي معادلة تفاضلية جزئية تكافئية، تُستخدم في نمذجة انتشار درجة الحرارة في قضيب معدني طوله l سم ، وتكتب المسألة والشروط الابتدائية الحدية كالتالي :

$$\frac{\partial u}{\partial t}(x,t)=\alpha^2\frac{\partial^2 u}{\partial x^2}(x,t), \qquad 0<x<l, \qquad t>0$$

تحت الشروط :

$$u(0,t)=0, \qquad u(l,t)=0, \qquad t>0$$

$$u(x,0)=f(x), \qquad 0\le x\le l$$

حيث إن α^2 يمثل معامل الانتشار ولإيجاد درجة الحرارة عند أي نقطة x وزمن t في قضيب يكون في درجة حرارة صفر عند النهايات $x=l$ و $x=0$ و بدرجة حرارة ابتدائية في القضيب $f(x)$. يتضمن الأسلوب الذي نستخدمه لتقريب حل هذه المسألة فروقاً محدودة ، وهي مشابهة للطريقة المستخدمة لحل المعادلات التفاضلية الجزئية الناقصية.

نختار أولاً ثابتي الشبكة h و k بشرط أن يكون $m=\frac{l}{h}$ عدداً صحيحاً. وبذلك نحصل على طريقة الفرق باستخدام متسلسلة تايلور ، ويمكن استخدام برنامج MATLAB لحل معادلة الحرارة بخوارزمية الفرق التراجعي Backward-difference method مع مراعاة شروط الاستقرار وكون خطأ القطع برتبة $O(k+h^2)$:

$$(1+2\alpha^2 k/h^2)u_{i,j}-(\alpha^2 k/h^2)(u_{i+1,j}+u_{i-1,j})=u_{i,j-1}$$

أو يمكننا استخدام طريقة كرانك– نيكلسون Crank-Nicholson method المستقرة بدون شروط وذات خطأ قطع من الرتبة $O(k^2+h^2)$:

$$\frac{u_{i,j+1}-u_{i,j}}{k}-\frac{\alpha^2}{2}\left[\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2}+\frac{u_{i+1,j+1}-2u_{i,j+1}+u_{i-1,j+1}}{h^2}\right]=0$$

مثال رقم (٥،٨)

قرب حل المعادلة التفاضلية الجزئية الآتية مستخدماً خوارزمية الفرق التراجعي:

$$\frac{\partial u}{\partial t}-\frac{\partial^2 u}{\partial x^2}=0, \quad 0<x<2, \quad 0<t;$$

بالشروط : $u(0,t)=u(2,t)=0, \quad t>0$

$$u(x,0)=\sin\frac{\pi}{2}x, \quad 0\le x\le 2.$$

باستخدام n=100 ، t=0.1 وm=10 وقارن النتائج بالحل الحقيقي:

$$u(x,t)=e^{-(\pi^2/4)t}\sin(\frac{\pi}{2})x.$$

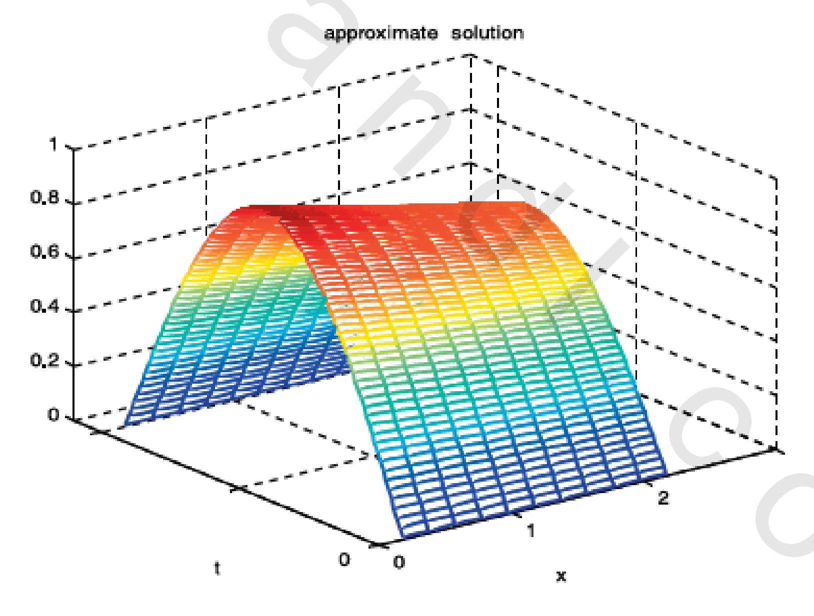
الحل:

الحل باستخدام برنامج *heat* (الملحق) ينتج الشكل رقم (٥،٧):

```
>> heat
The Parabolic equation of the form
d^2u/dt^2 - (alpha^2)* d^2u/dx^2 =0  0<x<l,  0<t<T
```

Subject to the boundary conditions
u(0,t)=T1
u(1,t)=T2   0<t<T
u(x,0)=f(x)   0<=x<=l

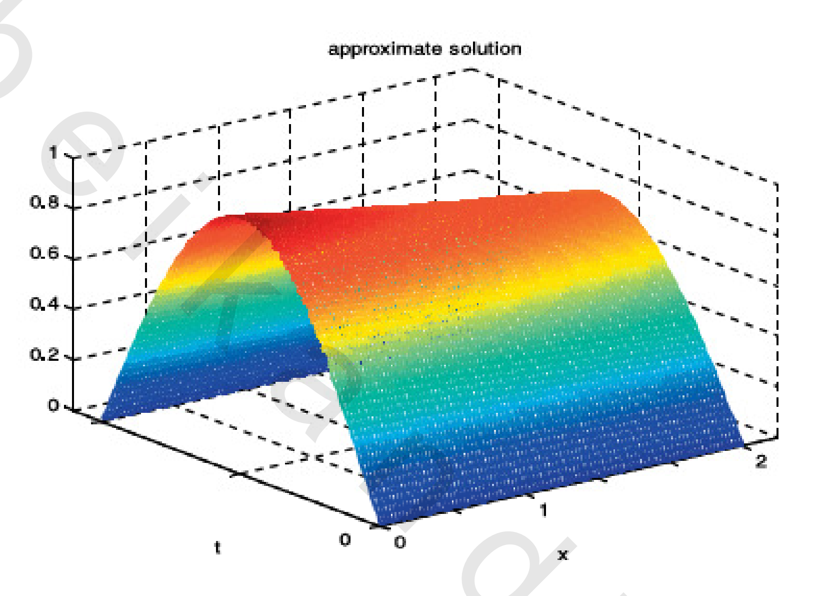
enter the number of grid sections for the x variable; m= 10
enter the number of grid sections for the t variable; n = 100
enter the end point of the range for x ; l= 2
enter the end point of the range for t ; T= .1
enter the exact solution e(x,t)=exp(-t.(pi^2/4)).*sin((pi/2).*x)*
enter the constant alpha = 1
enter the constant T1 = 0
enter the constant T2 = 0
*enter boundary condition u(x,0)=f(x)=sin((pi/2).*x)*



الشكل رقم (٥,٧). الحل الناتج لمعادلة الحرارة بقيم m= 100 , n=10 .

ولكن مع مراعاة شرط الاستقرار stability condition $\alpha^2(k/h^2) < 0.5$ أما عند استخدام كرانك- نيكلسون فإن الحل يكون أكثر دقة والطريقة مستقرة بدون شروط

كما في الشكل رقم (٥,٨)، باستخدام m= 100 و n=100 التي قد لا تكون مستقرة بخوارزمية الفرق التراجعي.



الشكل رقم (٥,٨). حل تقريبي لمعادلة الحرارة بقيم m= 100, n=100 بكرانك نيكلسون.

(٥,٦,٣) المعادلات التفاضلية الجزئية الزائدية

Hyperbolic Partial-Differential Equations

نتطرق للمعادلة التفاضلية الموجية كمثال لمعادلة تفاضلية جزئية زائدية وتعطى بالمعادلة التفاضلية الجزئية التالية:

$$\frac{\partial^2 u}{\partial t^2}(x,t)-\alpha^2\frac{\partial^2 u}{\partial x^2}(x,t)=0, 0<x<l, \qquad t>0$$

تحت الشروط :

$$u(0,t)=u(l,t)=0\ , \qquad t>0$$
$$u(x,0)=f(x), \qquad 0\le x\le l$$
$$\frac{\partial u}{\partial t}(x,0)=g(x), \qquad 0\le x\le l$$

ويستخدم برنامج MATLAB لحل المعادلة الموجية خوارزمية الفرق المحدود للمعادلة الموجية وهي بخطأ قطع برتبة $O(h^2+k^2)$ وشرط استقرار $\alpha k/h \le 1$ ، ويتم الحصول على طريقة الفرق باستخدام الفرق المركزي للمشتقات الجزئية الثانية لنصل الى معادلة الفرق الصريحة :

$$\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{k^2}-\alpha^2\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2}=0$$

مثال رقم (٥,٩)

أوجد حل المعادلة :

$$\frac{\partial^2 u}{\partial t^2}-\frac{\partial^2 u}{\partial x^2}=0, \quad 0<x<1, \qquad t>0$$
$$u(0,t)=u(1,t)=0, \qquad t>0$$
$$u(x,0)=\sin \pi x, \qquad 0\le x\le 1,$$
$$\frac{\partial u}{\partial t}(x,0)=0, \qquad 0\le x\le 1,$$

مستخدماً $m=10,\ n=10$ و $T=1$ وقارن بالحل الحقيقي للمعادلة $u(x,t)=\cos(\pi t)\sin(\pi x)$.

الحل :

نطبق البرنامج *wave* (الملحق) لينتج الحل التقريبي (الشكل رقم ٥,٩):

```
>> wave
The Hyperbolic equation of the form

d^2u/dt^2 - (alpha^2)*d^2u/dx^2 = 0  0<x<l,  0<t<T

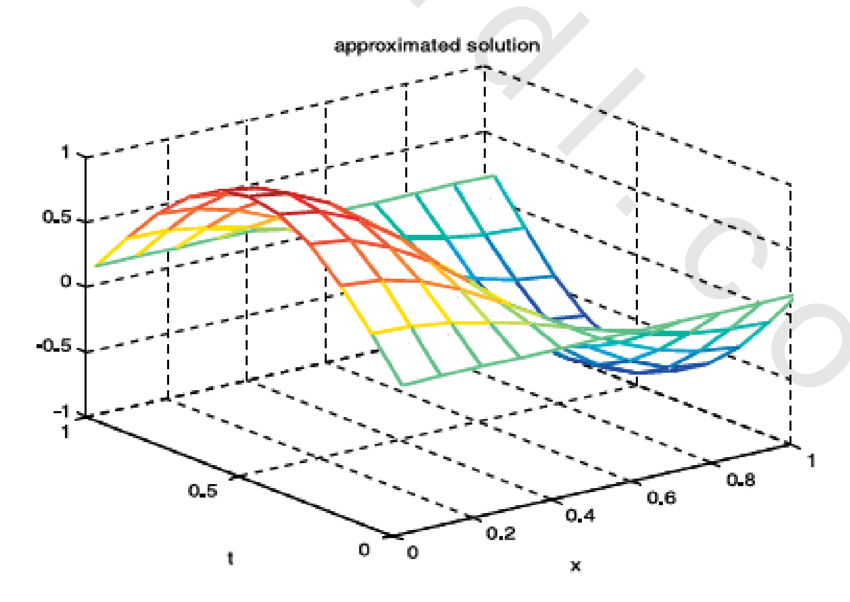
Subject to the boundary conditions

u(0,t)=u(l,t)=0       0<t<T

u(x,0)=f(x)

du/dt=g(x)       0<x<l

enter the number of grid sections for the t variable; n=  10
enter the number of grid sections for the x variable; m =  10
enter the end point of the range for x; l=  1
enter the end point of the range for t; T=  1
enter the constant alpha=  1
enter the boundary condition f(x)=sin(pi*x)
enter the boundary condition g(x)=0
enter the exact solution e(x,t)=cos(pi*t).*sin(pi*x)
```



الشكل رقم (٥,٩). حل تقريبي لمعادلة الموجه بقيم m= 10,n=10 .

الطرائق التي قُدمت في تطبيقات المعادلات التفاضلية الجزئية كانت مستقرة أو مشروطة الاستقرار وكلها طرائق صريحة ولكن توجد طرائق أخرى ضمنية ذات استقرار بدون شروط ويمكن رؤية دراسة لهذه الطرائق [22] .

(٥,٧) تمارين

حل المعادلات التفاضلية التالية باستخدام دالة جاهزة في MATLAB أو ببرنامج m-file :

١-

$$y'' = -(y')^2 - y + \ln x, \qquad 1 \le x \le 2$$
$$y(1) = 0,$$
$$y(2) = \ln 2$$

٢-

$$\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0, \quad 0 \le x \le \pi \qquad t > 0$$
$$u(0,t) = u(1,t) = 0, \qquad t > 0$$
$$u(x,0) = \sin x \qquad 0 \le x \le \pi$$

٣-

$$\frac{\partial u}{\partial t}(x,0) = 0, \qquad 0 \le x \le \pi$$
$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0, \quad 0 < x < 2, \qquad t > 0$$
$$0 \le x \le 2. \quad 0 < t; \; u(0,t) = u(2,t) = 0,$$

٤- أوجد الحل التقريبي بطريقة عددية مناسبة وقارن بالحل الفعلي.

أ) $$2\frac{d^2x}{dt^2}+4\left(\frac{dx}{dt}\right)^2-2x=\cos x$$

حيث إن $dy/dx=10$ و $x=0$ عندما $t=0$

ب) $-y''+y=x$ بحيث إن $0<x<1$ $y(0)=1$ و $y(1)=1+e^{-1}$

جـ) $0\le t\le 1$ $y(0)=0,$ $y'=\sin t+e^{-t}$

د) $0\le t\le 2$ $y(0)=0,$ $y'=t^2$

٥- استخدم الخوارزمية المناسبة لتقريب الحل :

أ) $0\le t\le 1$ $y(0)=0,$ $y'=1-y$

ب) $0\le t\le 5$ $y(0)=2,$ $y'=-y+t+1$

جـ) $1\le x\le 2$ $y(1)=0,\ y(2)=\ln 2$ $y''=-(y')^2-y+\ln x$

د) $0\le x\le 1$ $y(0)=0,\ y(1)=2$ $y''=4(y-x)$

الفصل السادس

استكمال وتقريب الدوال على MATLAB

عند وجود بيانات معينة حصيلة تجربة علمية، فإن تحليل هذه البيانات سيعطي معلومات عديدة، مثل : التنبؤ بالقيم المستقبلية، والحصول على قيم البيانات السابقة، أو قيم تقع بين نقاط البيانات المستخدمة. إن استخدام دالة تقريبية تمر بالبيانات لتسهيل تحليل البيانات يسمى بالاستكمال Interpolation. ولبرنامج MATLAB دورٌ باهرٌ في هذا المجال، فقد زوّد بأوامر جاهزة لاستكمال الدوال ورسمها بدقة فائقة في أي عدد من الأبعاد، كما يسهّل برمجة الدوال المتعلقة بالاستكمال، مما يجعله ينافس الكثير من البرامج المتخصصة في الرسم بسرعة ودقة فائقة. سنتطرق أيضاً في هذا الفصل لموضوع تقريب الدوال بدوال أبسط منها، مثل كثيرات الحدود والدوال المثلثية بنوعين من التقريب المتقطع والمتصل. وستتضمن الدراسة نظرية التقريب بدالة صريحة، والبحث عن الدوال الأكثر ملاءمة للتقريب باستخدام MATLAB.

(٦,١) استخدام كثيرة حدود للاستكمال Polynomial Interpolation

تستخدم كثيرات الحدود في التقريب، لأنها دوال متصلة، ولسهولة حساب كل من مشتقاتها وتكاملاتها. وتُعد كثيرات الحدود من أشهر الدوال وأكثرها استخداماً ،

خصوصاً في الاستكمال، وذلك لسهولة تطبيقها، وتكون على هذا الشكل:

$$p_n(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots\ldots + a_{n-1}x + a_n$$

حيث إن n عدد صحيح غير سالب و $a_0, a_1, \ldots\ldots, a_n$ أعداد حقيقية ثابتة، ومن أحد أسباب أهمية هذه الدوال هو أنها تقرّب الدوال المتصلة بانتظام. من كثيرات الحدود التي تستخدم في التقريب متسلسلة تايلور، وهي طريقة أساسية في التحليل العددي، ولكنها غير ملائمة للاستكمال لأن دقتها تتركز فقط في جوار نقطة التقريب. يتطلب الاستكمال كثيرة حدود تعطي تقريباً دقيقاً على فترة محددة وباستخدام معلومات من نقاط مختلفة.

سنعرض الاستكمال باستخدام كثيرات الحدود لاجرانج و نيوتن Newton, Lagrange polynomialsوكذلك تقريب كثيرات الحدود التجزيئية مثل دالة الشريحة التكعيبية للاستكمال Cubic Spline Interpolation مع تطبيقاتها على برنامج MATLAB.

(٦،١،١) استكمال كثيرة حدود لاجرانج

Polynomial Lagrange Interpolation

إذا كانت $x_0, x_1, \ldots, x_n$ أعداد مختلفة عددها n+1 وكانت $f(x)$ دالة قيمتها عند هذه الأعداد معروفة، فان طريقة Lagrange تعتمد على البحث عن كثيرة حدود P وحيدة، وبدرجة لا تزيد على n بحيث نحصل على:

$$p(x_i) = f(x_i) \qquad, i = 0,1,2,\ldots,n$$

عندها تكون كثيرة الحدود من الدرجة n بالشكل :

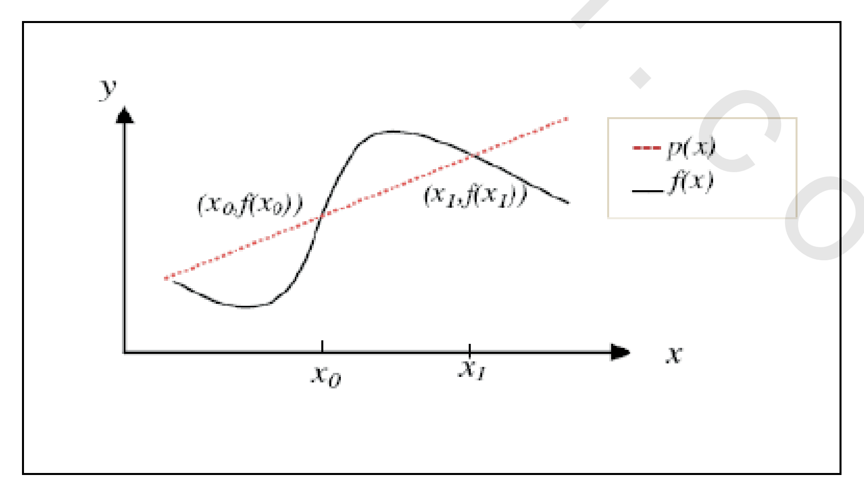
$$p_n(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + \dots + L_n(x)f(x_n)$$

بحيث إن :

$$L_k(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)(x_k-x_1)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)} = \prod_{i=0,k\neq i}^{n} \frac{x-x_i}{x_k-x_i}$$

وتسمى هذه الصيغة كثيرة حدود لاجرانج للاستكمال Lagrange Interpolation polynomial .

نبدأ بأسهل حالات الاستكمال ، وهي الاستكمال الخطي ، فلتكن $f(x)$ دالة معرفة عند النقطتين x_0, x_1 ونريد أن نكون كثيرة حدود $p_1(x)$ من الدرجة الأولى ، و تمر بالنقطتين $(x_0, f(x_0))$ و $(x_1, f(x_1))$. ويتضح من الشكل رقم (٦،١) أن الخط المستقيم هو أقصر منحنى يمر بين النقطتين السابقتين.



الشكل رقم (٦،١). الاستكمال الخطي.

ويمكن تمثيل هذا المستقيم بكثيرة حدود خطية كما يلي :

$$p(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1)$$

فإذا كانت :

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} \quad \text{و} \quad L_1(x) = \frac{x - x_0}{x_1 - x_0}$$

فإن كثيرة الحدود الخطية تأخذ الشكل التالي :

$$p_1(x) = L_0(x) f(x_0) + L_1(x) f(x_1)$$

التقريب بدوال لاجرانج يلعب دوراً مهماً في إيجاد صيغ التكامل العددية المختلفة التي تم عرضها في الفصل الرابع. يساعد برنامج MATLAB في إيجاد كثيرة حدود لاجرانج وحساب قيمتها وقيم L_i عند أي نقطة وذلك بتعريف الدالة *lin* [7]

```
% Lagrange Interpolation Method
function fi = lin (x,y,xi);
 dxi = xi-x; m = length(x); zeros(size(y));
L(1) = prod (dxi(2:m))/prod(x(1)-x(2:m));
L(m) = prod (dxi(1:m -1))/prod(x(m)-x(1:m -1));
for j = 2:m -1
   n = prod (dxi(1:j -1))*prod(dxi(j+1:m));
   d = prod (x(j)-x(1:j-1))*prod(x(j)-x(j+1:m));
   L(j) = n/d ;
end;
fi = sum (y.*L);
```

خوارزمية (٦,١).

مثال رقم (٦,١)

لتكن قيم الدالة معطاة في الجدول التالي ، أوجد كثيرة حدود لاجرانج لاستكمال هذه الدالة ، وأحسب قيمة الدالة عند 1.5 ؟

i	x_i	$f(x_i)$
0	0	1
1	1	1/2
2	2	1/3

الحل :

بإدخال البيانات نحصل على كثيرة حدود لاجرنج كما هي معرفة في ملف الدالة *lin* وبالأمر *syms xi* :

```
>> x = [0 1 2];
>> y = [1 1/2 1/3];
>> syms xi ;
>> p2 = lin (x,y,xi)
   p2 =
        1/2*(xi-1)*(xi-2)-1/2*xi*(xi-2)+1/6*xi*(xi-1)
```

ويمكن ترتيبها كما يلي :

```
>> p2 = expand (p2)
p2 =
 1/6*xi^2-2/3*xi+1
```

وهذا يعطي كثيرة الحدود :

$$p_2(x) = 0.1667x^2 - 0.6667x + 1$$

ولحساب قيمة كثيرة الحدود عند (1.5) ندخل في دالة *lin* كما يلي :

```
>> p2 = lin(x, y, 1.5)
   p2 =
          0.375
```

مثال رقم (٢, ٦)

أوجد كثيرة حدود لاجرانج بالنسبة للنقاط $x_0 = 0, x_1 = 1, x_2 = 2.5$

لاستكمال الدالة $f(x) = \frac{2}{x+2}$ عند $x = 2.3$.

الحل :

```
  >> x = [0 1 2.5];
>> y =  [1  0.6677  0.4444];
  >> syms xi;
  >> p2 = lin(x,y,xi)
  p2 = 2/5*(xi-1)*(xi-5/2)-4/9*xi*(xi-5/2) +16/135*xi*(xi-1)
```

وبعد التبسيط نحصل على :

$$p_2(x) = 0.074x^2 - 2.63x + 1$$

ولحساب قيمة $P_2(2.3)$:

```
p2 = lin(x, y, 2.3)
p2 =
   0.4548
```

وللمقارنة نوجد القيمة الفعلية $f(2.3)$:

$$f(2.3) = \frac{2}{2.3+2} = 0.4651$$

(٦,١,٢) استكمال كثيرة حدود نيوتن

Newton Polynomial Interpolation

تعتمد هذه الطريقة على تقريب المشتقات، وتسمى الفروق المقسومة Divided Difference وتعرف كما يلي:

الفرق المقسوم الصفري عند النقطة x_i :

$$f[x_i] = f(x_i)$$

والفرق المقسوم من الدرجة الأولى عند النقاط x_0 و x_1 :

$$f[x_0,x_1] = \frac{f[x_1]-f[x_0]}{(x_1-x_0)} = \frac{f(x_1)-f(x_0)}{(x_1-x_0)}$$

ومن الدرجة n-1 عند النقاط $x_0,x_1,\ldots,x_n$:

$$f[x_0,x_1,\ldots,x_n] = \frac{f[x_1,x_2,\ldots,x_n]-f[x_0,x_1,\ldots,x_{n-1}]}{(x_n-x_0)}$$

لنحصل على صيغة كثيرة حدود نيوتن لاستكمال الدالة $f(x)$ في النقاط $(x_i)_{i=0}^{n}$:

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0,x_1,\ldots x_k](x-x_0)\ldots(x-x_{k-1})$$

مثال رقم (٦,٣)

أ) احسب الفرق المقسوم للدالة $f(x) = x^3 + 7x^2 + 1$ عند النقاط $\{x_i\}_{i=0}^{4}$ المعطاة بالجدول التالي :

i	0	1	2	3	4
x_i	1	2	3	4	5

ب) احسب كثيرة حدود نيوتن لاستكمال الدالة عند (1.5).

الحل :

نطبق على MATLAB خوارزمية (٦,٢) ونقوم بإيجاد الفروق المقسومة و كثيرة الحدود وقيمتها عند أي نقطة بالدالة *divdif* [7] :

```
function D = divdif(x,y)
m = length (x);D = zeros(m,m);
D(:,1) = y(:);
for j = 2:m
   for i = j:m
      D(i,j) = (D(i,j-1)- D(i-1,j-1))/(x(i)-x(i-j+1));
   end ;
end
```

خوارزمية (٦,٢).

```
x = [1 2 3 4 5];
y = [9 37 91 177 301];
D = divdif(x,y)
D =
    9     0     0    0    0
   37    28     0    0    0
   91    54    13    0    0
  177    86    16    1    0
  301   124    19    1    0
```

نعوض القيم الموضحة بالجدول في صيغة نيوتن :

```
>> syms x;
>> p = 9+28*(x-1)+13*(x-1) *(x-2)+(x-1) *(x-2) *(x-3);
>> expand (p)
ans =
   1+7*x^2+x^3
```

أي أن كثيرة الحدود هي : $p(x) = x^3 + 7x^2 + 1$.

لإيجاد قيمة الدالة عند (1.5) :

```
>> subs (p, 1.5)
ans =
  20.125
```

كما يوجد لدى MATLAB دوال مخزنة وجاهزة للاستكمال الخطي ومنها *interp1* ولها الصيغة :

p = interp1(x, y, [xi],'linear')

مثال رقم (٤,٦)

لتكن الدالة $f(x) = x^{2.9}$ معرفة عند النقاط x = 1,2,...,5 نريد إيجاد تقريب للدالة عند x = 2.3, 3.8 :

```
>> x = 1:5;
>> y = x. ^(2.9);
>> p = interp1(x, y, [2.3, 3.8],'linear')
p =
  12.4822   49.4104
```

نجد أن الاستكمال يصبح أدق عندما نعتمد على نقاط أكثر ودرجة أعلى، فللحصول على كثيرة حدود من الدرجة الثالثة في MATLAB نستخدم دالة *interp1* ولكن باستخدام ('cubic') :

```
>> x = 1:5;
>> y = x.^(2.9);
>> p = interp1(x, y, [2.3, 3.8],'cubic')
p =
  11.0710   48.1599
```

مثال رقم (٦,٥)

استخدم صيغة الاستكمال المناسبة على الدالة $f(x)$ عند x = 1.23 وجدول القيم التالي ، مع المقارنة بين النتائج ، علماً بأن القيمة الحقيقية هي 3.4212 .

x_i	1	1.2	1.4	1.6	1.8
$f(x_i)$	2.7183	3.3201	4.0552	4.9530	6.0496

الحل :

```
>> x = [1 1.2 1.4 1.6 1.8];
>> y = [2.7183 3.3201 4.0552 4.9530 6.0496];
```

تقريب الدالة بصيغة (Lagrange) :

```
>> p = lin (x, y, 1.23)
p =
   3.4212
```

تقريب الدالة بصيغة (Newton) :

```
>> D = divdif(x,y)


D =
   2.7183        0        0        0        0
   3.3201   3.0090        0        0        0
   4.0552   3.6755   1.6663        0        0
   4.9530   4.4890   2.0337   0.6125        0
   6.0496   5.4830   2.4850   0.7521   0.1745

>> x = 1.23;
```

```
>>pd=2.7183+3.0090*(x-1)+1.6663*(x-1)*(x-1.2)+0.6125*(x-1)*(x-1.2)*(x-1.4)
pd =
   3.4211
```

تقريب الدالة بصيغة (*interp1*) الخطية :

```
>> p1 = interp1(x, y, [1.23],'linear')
p1 =
   3.4304
```

تقريب الدالة بصيغة (*interp1*) المكعبة :

```
>> pc = interp1(x, y, [1.23],'cubic')
pc =
   3.4210
```

نلاحظ أن القيم كلها متقاربة ، والأفضل هو تقريب لاجرانج.

(٦,٢) الشريحة التكعيبية للاستكمال Cubic Spline Interpolation

يتضح مما سبق أننا نحتاج إلى زيادة درجة كثيرة الحدود للحصول على تقريب أفضل ، ويمكن تجزئة مجال النقاط المعطاة إلى أجزاء أصغر ، وتطبيق الاستكمال في كل جزء ، وبذلك نتفادى عدم الاستقرار الذي قد يظهر في بعض الفترات الصغيرة من المجال . إحدى الطرق المشهورة لتجزئة فترة الاستكمال إلى فترات صغيرة وإيجاد كثيرات حدود مختلفة تسمى دالة كثيرة حدود تجزيئية Piecewise Polynomial Function ، ومن ثم فإن الدالة التقريبية هي عبارة عن تجميع من هذه الدوال ، و تُعرف بما يلي :

نعرف الدالة $f(x)$ بدالة كثيرة حدود تجزيئية إذا تحقق :

$f(x) = p_i(x)$ لكل $x \in [x_i, x_{i+1}]$ و $i = 0, 1, 2, \ldots, n-1$

وتسمى الأعداد $x_0, x_1, \ldots, x_n$ عقد الدالة $f(x)$ على الفترة $[a,b]$ بحيث إن :

$$a = x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n = b$$

و $p_0(x), p_1(x), \ldots, p_{n-1}(x)$ كثيرات حدود من الدرجة m على الأكثر.

بصورة عامة دالة الشريحة S(x) Spline functions من الدرجة m على الفترة [a,b]، حيث إن $a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$ هي دالة كثيرة حدود تجزيئية بحيث إن :

$$S(x) = p_i(x) \text{ على الفترة } [x_i, x_{i+1}]$$

حيث إن $p_i(x)$ كثيرة حدود من الدرجة m على الأكثر وتحقق :

$p_i^{(k)}(x_{i+1}) = p_{i+1}^{(k)}(x_{i+1})$ لكل و $i = 1, 2, \dots, n-2$ و $k = 1, 2, \dots, m-1$.

وتسمى $x_1, \dots, x_{n-1}$ بالعقد الداخلية. والأكثر شيوعاً من بين هذه الدوال هي الدالة الشريحة التكعيبية من الدرجة الثالثة.

(٦,٢,١) دالة الشريحة التكعيبية Cubic Spline function

لتكن f دالة معرفة على الفترة [a , b] حيث إن $a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$ ، فنقول إن دالة الشريحة S(x) هي دالة شريحة تكعيبية و تستكمل f عند العقد $x_0, x_1, \dots, x_{n-1}$ إذا كانت تحقق :

- $S_i(x_i)$ وهي كثيرة حدود من الدرجة الثالثة على الفترة $[x_i, x_{i+1}]$ لكل $i = 0, 1, \dots, n-1$.
- $S(x_i) = f(x_i)$ لكل $i = 0, 1, \dots, n$.
- $S_{i+1}(x_{i+1}) = S_i(x_{i+1})$ لكل $i = 0, 1, 2, \dots, n-2$.
- $S'_{i+1}(x_{i+1}) = S'_i(x_{i+1})$ لكل $i = 0, 1, 2, \dots, n-2$.
- $S''_{i+1}(x_{i+1}) = S''_i(x_{i+1})$ لكل $i = 0, 1, 2, \dots, n-2$
- أحد الشروط الحدية التالية يكون متحققاً :

i) $S''(x_0) = S''(x_n) = 0$ (حد حر)

ii) $S'(x_0) = f'(x_0)$ $\quad S'(x_n) = f'(x_n),$ (حد مقيد)

الشريحة التكعيبية تسمى الشريحة الطبيعية عندما يتحقق شرط الحد الحر، ولكن الشروط المقيدة الحدية تؤدي إلى تقريب أكثر دقة، لأنها تحتوي على معلومات أكثر حول الدالة. ويمكن كتابة الدالة التكعيبية على الشكل التالي:

$$S_i(x) = a_i(x-x_i)^3 + b_i(x-x_i)^2 + c_i(x-x_i) + d_i$$

حيث إن a_i, b_i, c_i, d_i لكل $i = 0, 1, 2, \dots, n-1$ هي الثوابت التي يتم تعيينها.

أما على MATLAB فإنه يوجد دوال *interp1* و *spline* للتعامل مع الشريحة التكعيبية وفق المثال رقم (٦,٦).

مثال رقم (٦,٦)

لتكن قيم x و y التالية $x = \{0, 1, 2, 3, 4\}$ و $y = \{3, 1, 0, 2, 4\}$

أوجد قيمة الدالة y عند $x = 1.5$ باستخدام الشريحة التكعيبية (cubic spline) مع رسم الدالة.

الحل:

نوجد قيمة الدالة y عند $x = 1.5$ كما يلي:

```
>> x = [0 1 2 3 4 ];
>> y = [3 1 0 2 4 ];
>> xval = 1.5;
>> yval = spline(x,y,xval)
yval =
   0.1719
```

ويمكن الوصول لنفس النتيجة بالأمر التالي :

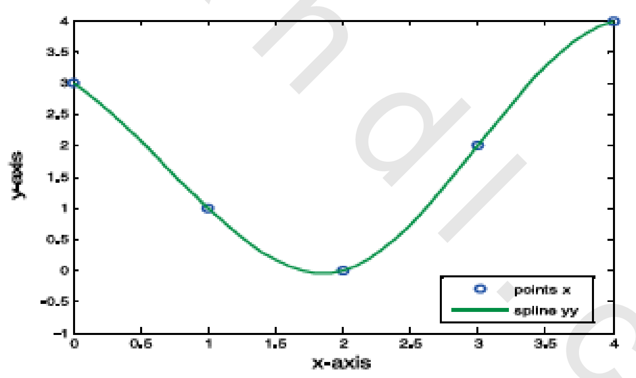
```
YY = ppval(spline(X,Y),XX)
```

أما استخدام *interp1* فيتم بالخطوات التالية :

```
>> p = interp1 (x,y ,[1.5], 'spline')
p =
   0.1719
```

ونرسم الشكل رقم (٦,٢) كما يلي :

```
>> xx = 0:.1:4;
>> yy = spline (x,y,xx);
>> plot (x,y,'o',xx,yy) ; axis ([0 4 -1 4 ]);
>> xlabel ( 'x-axis' ) ; ylabel ('y-axis')
```



الشكل رقم (٦,٢). رسم كثيرة الحدود الناتجة عن استكمال الشريحة المكعبة .

(٦,٣) تقريب بطريقة أصغر المربعات Least squares approximation

لدراسة إيجاد دالة تُمثل كمية كبيرة نسبياً من البيانات تحوي على أخطاءً حصلت في أثناء جمعها ، فإن عملية التقريب تتعلق بإيجاد دالة مطابقة للبيانات ،

واختيار أحسن دالة لتمثيل هذه البيانات. وينصح بطرائق أصغر المربعات المتقطعة عندما تكون الدالة معينة ببيانات نقطية قد لا تمثل قيماً حقيقيةً لدالة. في هذه الحال يمكن لأصغر المربعات أن يأخذ شكلاً خطياً أو بدرجة كثيرة حدود أخرى، أو دالة أُسية أو مثلثية. و ليس من المهم أن نحصل على كثيرة حدود ذات درجة عالية، ولكن الأهم أن تكون انسيابية وبسيطة، حتى لو لم تطابق تماماً البيانات بسبب الأخطاء في إجراء تجميع للبيانات، مثلاً إذا جمعت بيانات لتجربة ما، و كانت كما يلي :

X_i	1	2	3	4	5	6	7	8	9	10
Y_i	1.3	3.5	4.2	5.0	7.0	8.8	10.1	12.5	13.0	15.6

ونريد إيجاد دالة خطية تمر بهذه القيم (x_i , y_i) باستخدام نظرية تقريب أصغر مربعات، فإننا نحتاج لتعريف مجموع نسبة الخطأ ، أو ما يسمى بالانحراف المطلق Absolute deviation وهو :

$$\sum_{i=1}^{m} |y_i - (ax_i + b)|$$

حيث إن $(ax_i + b)$ هي القيمة على الخط المستقيم التقريبي و y_i هي قيمة y المعطاة عند i ، و m هي عدد النقاط المعطاة. إن طريقة أصغر مربعات تعتمد على إيجاد أقل قيمة للانحراف المطلق حيث إنه يُعد أحسن تقريب خطي ، لذلك يتطلب إيجاد قيم a و b حتى نتمكن من تقليل نسبة الخطأ إلى الحد الأدنى. ومن ثم حساب :

$$\varepsilon = \sum_{i=1}^{m} [y_i - (ax_i + b)]^2 \ldots\ldots\ldots\ldots(*)$$

طريقة أصغر مربعات تكون أكثر إجراء ملائم لتحديد أحسن تقريب خطي ، حيث إن هذا التقريب يعتمد على كل النقاط على الخط المستقيم التقريبي وخارجه ، و لا يسمح للنقاط الخارجة بالسيطرة الكاملة على التقريب. نريد الآن تقليل المقدار إلى

الحد الأدنى بالنسبة للمتغيرين a و b ، ولكي نحصل على ذلك لا بد أن يكون :

$$0 = \frac{\delta}{\delta a}\sum_{i=1}^{m}[y_i - (ax_i + b)]^2 = 2\sum_{i=1}^{m}(y_i - ax_i - b)(-x_i)$$

$$0 = \frac{\delta}{\delta b}\sum_{i=1}^{m}[y_i - (ax_i + b)]^2 = 2\sum_{i=1}^{m}(y_i - ax_i - b)(-1)$$

نبسط هاتين المعادلتين إلى المعادلتين العاديتين normal equations :

$$a\sum_{i=1}^{m} x_i^2 + b\sum_{i=1}^{m} x_i = \sum_{i=1}^{m} x_i y_i$$

$$a\sum_{i=1}^{m} x_i + b.m = \sum_{i=1}^{m} y_i$$

الحل لهذا النظام يكون :

$$a = \frac{m\left(\sum_{i=1}^{m} x_i y_i\right) - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i}{m\left(\sum_{i=1}^{m} x_i^2\right) - \left(\sum_{i=1}^{m} x_i\right)^2}$$

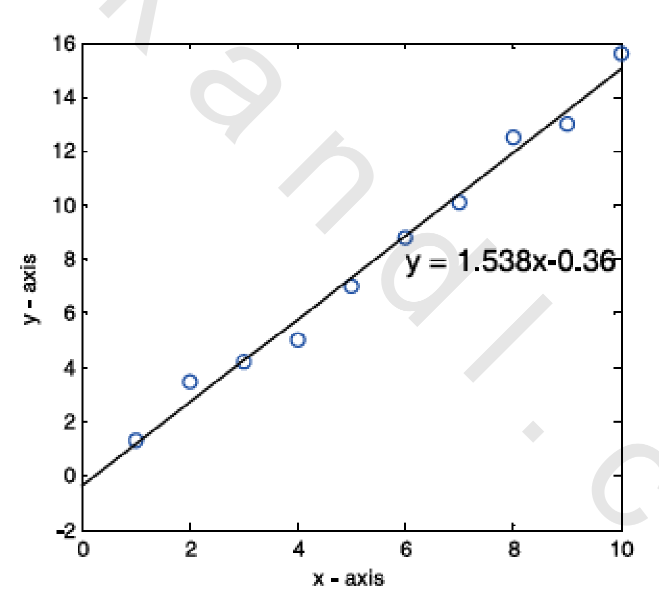
$$b = \frac{\left(\sum_{i=1}^{m} x_i^2 \sum_{i=1}^{m} y_i\right) - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} x_i y_i}{m\left(\sum_{i=1}^{m} x_i^2\right) - \left(\sum_{i=1}^{m} x_i\right)^2}$$

ونطبق ذلك على البيانات لإيجاد أصغر مربعات لتقريب البيانات إلى خط مستقيم. حيث إن قيمتي a و b يمكن حسابهما كما يلي :

$$a = \frac{10(572.4) - 55(81)}{10(385) - (55)^2} = 1.538$$

$$b = \frac{385(81) - 55(572.4)}{10(385) - (55)^2} = -0.360$$

يمكننا الرسم بعد أن حصلنا على القيم التقريبية بوساطة أصغر مربعات لنقط البيانات، كما في الشكل رقم (٦,٣):



الشكل رقم (٦,٣). تمثيل لكثيرة الحدود الخطية الناتجة عن أصغر مربعات.

وبشكل عام لتقريب مجموعة من البيانات $\{(x_i, y_i) : i = 1, 2, \ldots, m\}$ إلى كثيرة حدود باستخدام أصغر

المربعات تقوم بنفس الأسلوب، أي يتطلب منا اختيار الثوابت $a_0 , a_1 , \ldots , a_n$ حتى نخفض خطأ أصغر المربعات إلى الحد الأدنى. طريقة إيجاد كثيرة الحدود من الدرجة n لنقاط بيانات x_i بطريقة أصغر مربعات على MATLAB تتم باستخدام دالة تسمى *polfit* ، حيث تعطينا معاملات كثيرة الحدود $p_n(x)$. كذلك يمكننا إيجاد قيم لكثيرة الحدود تساعد في رسمها بدقة بدالة تسمى *polyval*. وهناك خوارزمية (٦،٣) للشكل العام لأصغر مربعات General Least Squares وتدعى *fgenfit1* [15] حيث يمكن استعمالها في تقريب أي مجموعة من البيانات إلى كثيرة حدود مناسبة للبيانات.

```
function c = fgenfit1(func,x,y)
n = length(y);
[p,jj] = feval (func,x(1));
A = zeros(p,p); b = zeros (p,1);
for i = 1:n
  [jj,f] = feval (func,x(i));
    for j = 1:p
       for k = 1:p
           A(j,k) = A(j,k)+f(j)*f(k);
end
        b(j) = b(j)+y(i)*f(j);
    end;
end
c=A\b;
```

خوارزمية (٦،٣).

مثال رقم (٦،٧)

أوجد تقريباً مناسباً للدالة $y = \sin\{1/(x + 0.2)\} + 0.2x$ عند البيانات التالية :

$$xs = [0:0.05:0.25 \quad 0.25:0.2:4.85]$$

وبنسبة خطأ 0.06 ، مع مقارنة أمر *fgenfit1* بكثيرة الحدود من الدرجتين الثالثة والخامسة *polfit* .

الحل :

نوجد كثيرة الحدود بأمر *fgenfit1* وذلك بتعريف الدالة *f3* في m-file :

$$z_1 = 1 \quad , \quad z_2 = \sin\{1/(x + 0.2)\} \quad , \quad z_3 = x$$

```
function [df,z] = f3(xs)
[j,n] = size(xs);df = 3;  z = zeros (df,n);
z(1,:) = ones (1,n) ;  z (2,:) = sin (ones(1,n)./(xs+0.2));
z (3,:) = xs;
```

نعرف البيانات مع نسبة الخطأ كما يلي :

```
>> xs = [0:.05:.25  .25:.2:4.85];
>> us = sin (ones(size(xs))./(xs+.2))+.2*xs +0.06*randn(size(xs));
```

نتفذ أمر *fgenfit1* لنحصل على معاملات كثيرة الحدود كما يلي :

```
>>  xx=0:.05:5;
>>  c = fgenfit1('f3',xs,us)
c =
-0.0049
1.0716
0.1943
```

فيمكن كتابة كثيرة الحدود كما يلي :

$$p_2(x) = -0.0049 + 1.0716x + 0.1943x^2$$

نوجد قيم كثيرة الحدود الناتجة، و رسمها كما يلي :

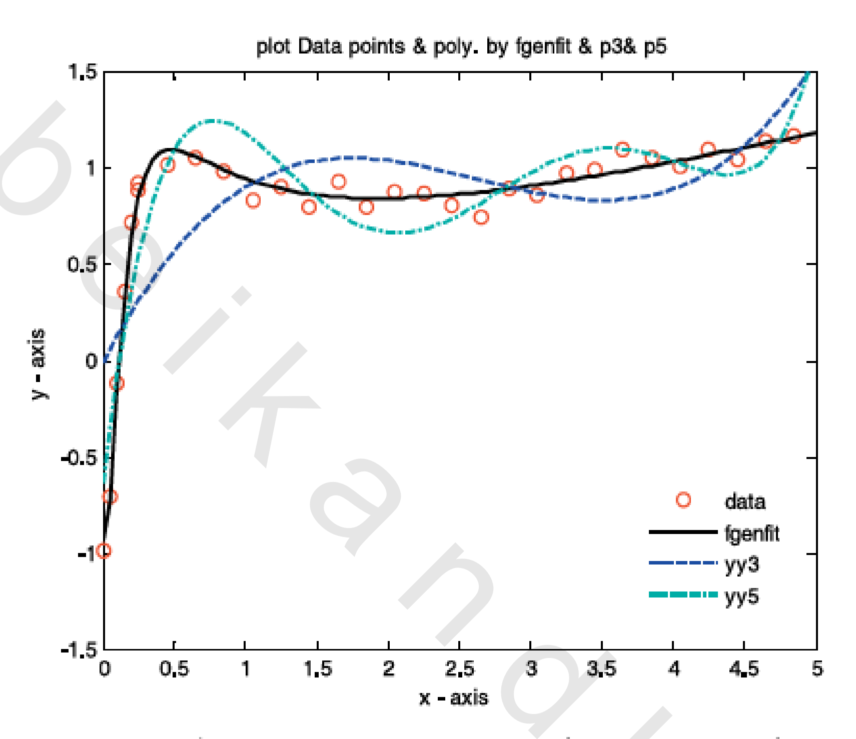
```
[j,p] = feval ('f3',xx);
 yy = c'*p;
 plot (xs,us,'o',xx,yy)
```

نرسم كثيرتي الحدود p_3 و p_5 على نفس الرسم بأمر *polyfit* الشكل رقم (٦,٤) كما يلي :

```
 p3 = polyfit (xs,us,3)
p3 =
   0.0787   -0.6263    1.4811   -0.0368
 yy3 = polyval (p3,xx);
 plot (xx,yy3,'--')
hold on
 p5 = polyfit (xs, us, 5)
p5 =
   0.0505   -0.6821    3.3577   -7.2294    6.3426   -0.6581
 yy5 = polyval (p5, xx,':');
 plot (xx,yy5)
 axis ([0 5 -1.5 1.5]); xlabel ('x - axis ');  ylabel ('y - axis');
title (' plot Data points & poly. by fgenfit & p3& p5'); legend
('data','fgenfit67','yy3','yy5');
```

نجد مما سبق أنه يمكن رسم منحنى تقريبي لأي دالة بأمر *polfit* حيث إنه من الممكن التحكم بدرجة كثيرة الحدود، ولكن لا يعطينا نتائج مثالية. وأما في أمر *fgenfit1* فإنه مبرمج على أن يختار أحسن درجة لكثيرة الحدود تناسب البيانات المعطاة، وبذلك نحصل على منحنى تقريبي مثالي، كما لاحظنا أن نسبة الخطأ بين كثيرات الحدود الناتجة والدالة المعطاة y هي الأقل بأمر *fgenfit1*.

من المناسب أحياناً أن نفترض أن المعطيات مرتبطة فيما بينها أُسّيا، وهذا يعني أن الدالة المقربة هي من الشكل $y=be^{ax}$ ، $y=bx^{a}$ أو دالة كسرية ، فيمكن استعمال أوامر MATLAB الجاهزة للتعامل معها ، كما في المثال رقم (٦,٨).



الشكل رقم (٦,٤). تمثيل لكثيرات الحدود P5 & P3 & P2 الناتجة عن أصغر مربعات

مثال رقم (٦,٨)

لدينا البيانات التالية :

x= [0:0.25:3]
y= [6.3806 ,7.1338 ,9.1662 ,11.5545 ,15.6414 ,22.7371 ,32.0696 ,47.0756 ,73.1596, 111.4684, 175.9895 ,278.5550 ,446.4441]

١– أوجد كثيرات الحدود على النمط الاتي :

1.$f_1(x) = a + be^x + ce^{2x}$ (استعمل أمر fgenfit)

2.$f_2(x) = a + b/(1+x) + c/(1+x)^2$ (استعمل أمر fgenfit)

3.$f_3(x) = a + bx + cx^2 + dx^3$ (استعمل أمر polyfit)

٢- ارسم الدوال الثلاث والنقاط (y). أيٌّ منها يمثل أقرب منحنى للنقاط المعطاة؟ علماً بأن القيم أخذت من $f(x) = 3 + 2e^{x} + e^{2x}$ مع إضافة تغييرٍ صغير عشوائي عليها.

الحل :

نعرف الدوال *f11(x)* و *f22(x)* على شكل m-file :

```
function [d,z]=f11(x)
 [j,n]=size(x);d=3;z=zeros(d,n);
 z(1,:)=ones(1,n);z(2,:)=exp(x);z(3,:)=exp(2.*x);
```

```
function [d,z]=f22(x)
 [j,n]=size(x);d=3;z=zeros(d,n);
z(1,:)=ones(1,n);z(2,:)=(ones(1,n)./(1+x));z(3,:)=(ones(1,n)./((x+1).^2))
```

أما طريقة إيجاد الثوابت ورسم كثيرات الحدود *f1* و *f2* و *f3* مع النقاط y ، فهي :

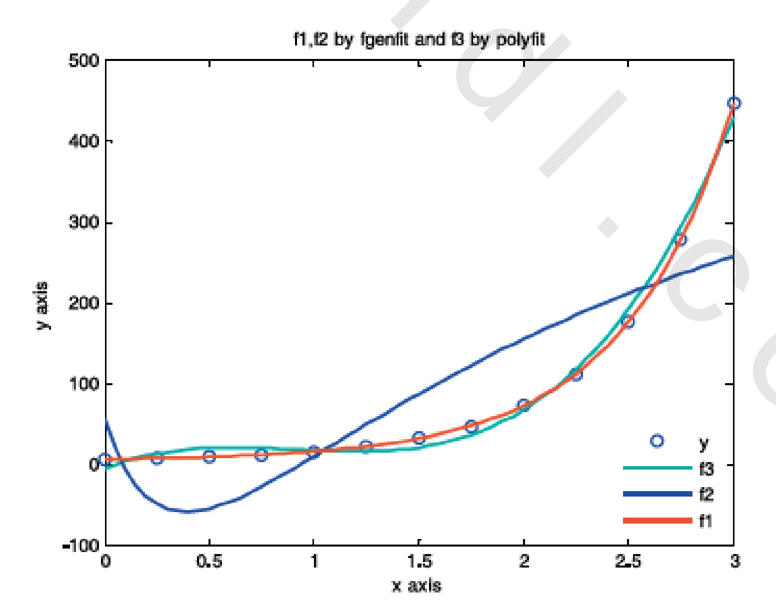
```
x = [0:0.25:3];
y = [6.3806 7.1338 9.1662 11.5545 15.6414 22.7371 32.0696 47.0756
73.1596   111.4684 175.9895 278.5550 446.4441];
xx = [0:.05:3];
f1 = fgenfit('f11',x,y)
[j,p] = feval('f11',xx); yy1 = (f1)'*p;
f2 = fgenfit1 ('f22',x,y)
[j,p] = feval('f22',xx); yy2 = (f2)'*p;
xx = [0:.05:3];
p = polyfit(x,y,3)
yy3 = polyval(p,xx);
```

```
plot (x,y,'o',xx,yy1,xx,yy2,xx,yy3) ;   xlabel ('x axis'); ylabel ('y axis');
title ('figure(1) f1,f2 by fgenfit and f3 by polyfit'); legend ('y','yy1','yy2','yy3')
```

f1 =
3.1276
1.9811
1.0000
f2 =
*1.0e + 003 **
0.6851
-2.0722
1.4438
f3 =
47.3747 -128.3479 103.4153 -5.2803

نجد في الشكل رقم (٦،٥) أن كثيرات الحدود الثلاث هي :

i) $f_1(x) = 3.1276 + 1.9811e^x + e^{2x}$
ii) $f_2(x) = 685.1 - 2072.2/(1+x) + 1443.8/(1+x)^2$
iii) $f_3(x) = 47.3747\, x^3 - 128.3479\, x^2 + 103.4153\, x - 5.2803$



الشكل رقم (٦،٥). رسم الدوال f3 & f2 & f1 مع نقاط الدالة الأصلية.

(٦,٤) تحليل فورير Fourier Analysis

(٦,٤,١) متسلسلات فورير Fourier Series

يمكن أن يُنتج تقريب عدد كبير من النقاط الموزعة بانتظام بواسطة كثيرات حدود مثلثية نتائج دقيقةً جداً ، وهي طريقة التقريب المناسبة للاستخدام في مجالات كثيرة، كالأجهزة التي تحتوي على مصفيات عددية، أو في الموجات التي يتم استقبالها في الهوائيات، وكذلك في الميكانيكا وغيرها من المجالات. إلا أن هذه الطريقة لم تطّبق كثيراً حتى منتصف عام ١٩٦٠م، وذلك بسبب الحسابات العديدة اللازمة لتعيين الثوابت في عملية التقريب، إذ إن استكمال عدد 2m من النقاط بالطرائق الحسابية المباشرة يتطلب حوالي $(2m)^2$ عملية ضرب، ومثلها عملية جمع. وفي عام ١٩٦٥ م نُشرت طريقة جديدة لحساب ثوابت كثيرة الحدود الاستكمالية المثلثية، و بعدد حسابات أقل ، بشرط أن نختار m بطريقة مناسبة. وأحدثت هذه الطريقة ثورة في استخدامات كثيرات حدود الاستكمال المثلثية وعرفت بخوارزمية تحويل فورير السريع .

وتعرف متسلسلات فورير للدالة $f(x)$ الدورية ودورتها 2π كما يلي :

$$f(t)=\frac{c_0}{2}+\sum_{n=1}^{\infty}c_n\cos(nt)+\sum_{n=1}^{\infty}s_n\sin(nt)$$

حيث إن الثوابت S_n ، C_n تعرف كما يلي :

$$c_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(t)\cos(nt)dt$$

$$s_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(t)\sin(nt)dt$$

وفي MATLAB فإنه من الممكن إيجاد ثوابت متسلسلة فورير ورسمها بدقة ووضوح.

مثال رقم (٦,٩)

أوجد متسلسلة فورير للدالة $f(t)= \sin(t)\cos(t)^2$ على الفترة $[-\pi,\pi]$ والثوابت S_n و C_n بالرسم.

الحل :

نعرف الدالة بوساطة m-file كما يلي :

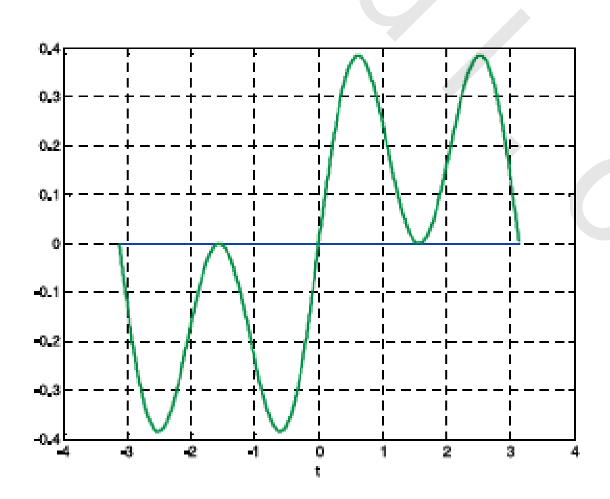
```
function F = fun (T);
F = sin (T).*(cos (T).^2);
```

ونعرف الدالتين *fc* و *fs* على شكل m-file كما يلي :

```
% c
function Y = fc (T);
global n
Y = fun(T).*cos (n.*T)
% S
function Y= fs (T);
global n
Y= fun (T).*sin (n.*T)
```

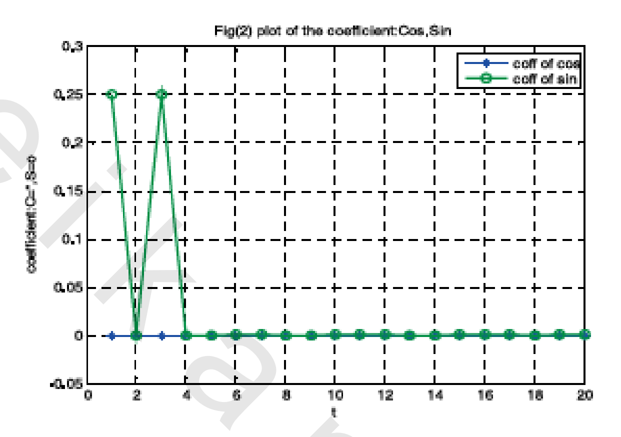
نستخدم للتكامل دالة *quad8* على الفترة [π , π−] ونجمع الحدود لتكوين كثيرة حدود فورير النهائية y. نرسم الدالة الأصلية f و الدالة الجديدة y، وكذلك نرسم المعاملات بالأوامر التالية (الشكل رقم ٦,٦) :

```
global n;T = -pi:.01:pi ; Y = 0*T;
for n1= 0:20
   n = n1;
   c = quad8 ('fc',-pi,pi)/pi;
   if n == 0,c = c/2 ;end
   s = quad8 ('fs',-pi,pi)/pi;
     Y = Y + c*cos (n*T) + s*sin (n*T);
  figure (1),plot (T,Y,T, fun (T) ), grid on
         if n > 0
     N (n) = n ; C (n) = c ; S (n) = s;
                              end
end
figure (2), plot (N,C,'*-',N,C,N,S, 'o-',N,S), grid on
```

الشكل رقم (٦،٦). رسم متسلسلة فورير للدالة sin(t)^2* cos(t)^2 .

من خلال الشكل رقم (٦,٧) نجد أن S_1 , S_3 تساويان 0.25 وأما باقي الثوابت تكاد تكون صفرية.



الشكل رقم (٦,٧). رسم ثوابت متسلسلة فورير Sn و Cn .

(٦,٤,٢) تحويلات فورير Fourier Transforms

يُعد تحويل فورير من أهم وأكثر التوابع استخداماً ، حيث يساعدنا على تحليل البيانات لمعرفة التردد، وهذا مفيد جداً لمعرفة المحتوى الترددي للبيانات وتستخدم المصفوفات للتعبيرعن صورة أو إشارة صوتية أو غيرها. نعرف تكامل فورير للدالة *f(t)* الدورية، والتي طول دورتها L ، على الشكل :

$$f(t) = \int_0^{\infty} c(v)\cos(2\pi v t)dv + \int_0^{\infty} s(v)\sin(2\pi v t)dv$$

حيث إن $\nu = \frac{n}{L}$ التردد لدالة متغيرة مع الزمن ويعبر عنه أيضاً بالصيغة المركبة الأسّية :

$$f(t) = \int_{-\infty}^{\infty} g(\nu)\exp(i\,2\pi\nu t)d\nu$$

وبشكل مماثل يمكن التعبير عن g(v) كما يلي :

$$g(\nu) = \int_{-\infty}^{\infty} f(t)\exp(-i\,2\pi\nu t)dt$$

أي أن :

$$g(\nu) = \int_{-\infty}^{\infty} f(t)\{\cos(-2\pi\nu t) + i\sin(-2\pi\nu t)\}dt$$

$$= \int_{-\infty}^{\infty} f(t)\cos(2\pi\nu t)dt - i\int_{-\infty}^{\infty} f(t)\sin(2\pi\nu t)dt \,........(*)$$

فعندما تكون الدالة *f(t)* زوجية ، فإن التكامل الثاني يكون صفراً ، وتتحول الدالة *g(v)* إلى دالة حقيقية. أما إذا كانت الدالة *f(t)* فردية ، فإن التكامل الأول يتلاشى وتتحول الدالة *g(v)* إلى دالة تخيلية.

(٣,٤,٦) تحويلات فورير المتقطعة (DFT) Discrete Fourier Transforms

لتحليل موجة نبضية من حيث التذبذب البسيط على سلسلة من الفترات الزمنية، لتكن $t = t + k\,\delta t$ حيث إن $k = 1, 2, \ldots$ و δt هي الفترة الزمنية بين قياسين متتاليين للزمن، نستخدم تحويلات فورير المتقطعة. ويمكن التعبير

عن متسلسلة فورير بالشكل التالي :

$$g(v_m) = \int_{-\infty}^{\infty} f(t)\exp(-i\,2\pi v_m)dt \cong \delta t \sum_{k=1}^{n} f(t_k)\exp(-i\,2\pi v_m t_k)$$

وفي كل مجموع يعطينا قيمة لـ $g(v)$ عند التردد v_m . تحويل فورير البسيط Simple Fourier Transform يعرف بالدالة *sft* (خوارزمية ٦,٤) التي تنفذ تحويل فورير البسيط على MATLAB [6] كما يلي ، لاستخدامها في العديد من الأمثلة.

```
% Simple Fourier Transform
function G = sft(T,F,N);
            dt= T(2)-T(1);
n = length(N);
for k =1:n
   G(k)= dt*sum (F.*exp (-i*2*pi*N(k)*T));
end
```

خوارزمية (٦,٤).

والمثال رقم (٦,١٠) يوضح استخدام MATLAB في رسم كثيرة الحدود الناتجة عن تحويل فورير بسيط لدالة الجا Transform of a Sine Function .

مثال رقم (٦,١٠)

أخذت عينة تحتوي على 1001 نقطة من البيانات ، ليكن لدينا الثوابت التالية :

. a = (2/n) و n = 2

ارسم دالة تحويل فورير للدالة (sine) وكذلك المعاملات (b_n).

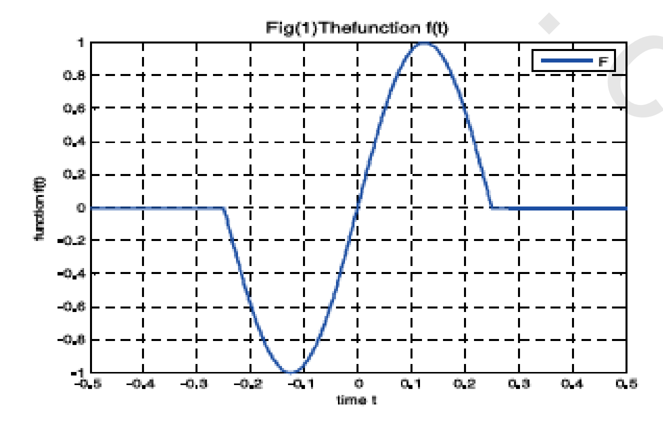
الحل :

في البداية نعرف الدالة التالية على شكل m-file وهي تساعد في تعريف الدالة F الوحدة النبضية [6] :

```
% Rectangular Pulse
function up = unitpulse (x,y,z);
up = unitstep (x-y) - unitstep(x-z);
```

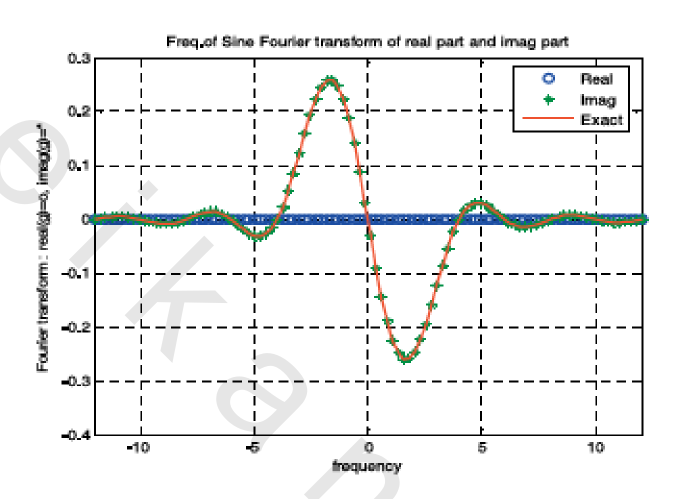
والآن لإيجاد تحويل فورير للدالة (sine) نقوم بتنفيذ الأوامر وينتج الشكل رقم (٦،٨) :

```
nn = 1001;
 n = 2;  a =2/n; w = 2*pi*n;
t = 2*a;
dt = 2*t/(nn-1);
T = -t :dt :t;
F = sin (w*T).*unitpulse (T,-a,a);
figure (1), plot (T,F);grid on ;
 N = linspace (-12,12,100);
G = sft (T,F,N);
W = 2*pi*N;
Gm =(sin (a*(W+w))./(W+w) - sin(a*(W-w))./(W-w));
figure (2), plot(N, real (G), 'o', N, imag(G), '*', N, Gm), grid on
```



الشكل رقم (٦،٨). رسم لدالة فورير (sine) لدورة واحدة .

نلاحظ من الشكل رقم (٦،٩) أن المعاملات (a_n) الحقيقية كلها صفرية ، و ذلك لأن دالة (sine) عبارة عن دالة فردية.



الشكل رقم (٦،٩). رسم الثوابت الحقيقية (O) و التخيلية (*) فورير sine .

(٦،٤،٤) تحويلات فورير العكسية Inverse Transform of Fourier

قد نحتاج إلى الطريقة العكسية ، وذلك الحصول على الدالة *f(t)* من *g(v)* معلومة ، وهذا ما يسمى بتحويلات فورير العكسية **Inverse Transform of Fourier**، ويتم ذلك في MATLAB بالدالة (خوارزمية ٦،٥) *isft* ونلاحظ أن الفرق بينها وبين *sft* هو التبديل بين متغيرات المدخلات والمخرجات [6] .

```
Function f=isft(n,g,t);
D=n(2)-n(1);
nt=length(t);
for k=1:nt
F(k)=D*sum(g.*exp(i*2*pi*n*t(k)));
end
```

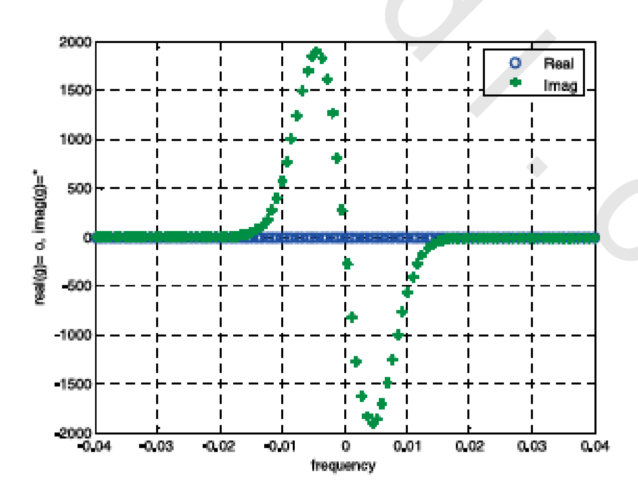
خوارزمية (٦،٥).

مثال رقم (٦,١١)

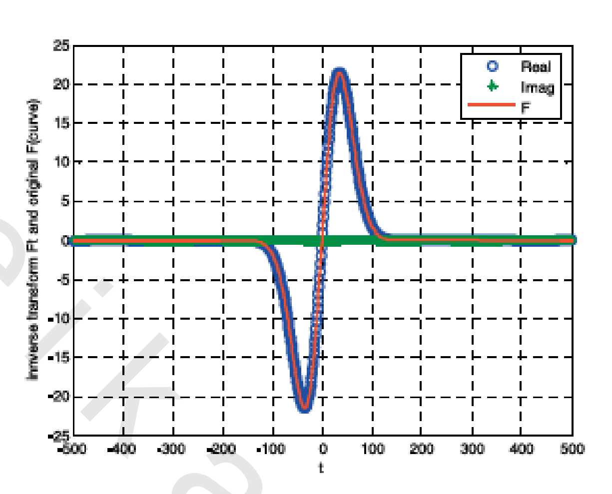
ليكن لدينا الدالة $f(t) = te^{-t^2/a^2}$ طبّق عليها تحويل فورير لتحصل على الدالة G ، استخدم الدالة G لإيجاد تحويل فورير العكسي (ft) وحيث إن a=50.

الحل :

نطبق دالتي *isft* و *sft* ونلاحظ من الرسم الأول (الشكل رقم ٦,١٠) المعاملات التخيلية للدالة (G) وهي التي نريد إيجاد فورير العكسي لها ، والتي تشبه تماماً الدوال الفردية. كما نجد من الرسم في الشكل رقم (٦,١١) انطباق (ft) دالة التحويل العكسي على الدالة الأصلية *f* ويدل الرسم أيضاً على أن (ft) عبارة عن دالة ذات قيم حقيقية.



الشكل رقم (٦,١٠). رسم الثوابت الحقيقية (O) و التخيلية (*) لدالة فورير G(t).



الشكل رقم (٦,١١). رسم الثوابت الحقيقية (O) لدالة فورير العكسي (ft) والدالة (f) .

(٦,٤,٥) دوال رمزية لتحويل فورير

توجد دوال جاهزة لتحويل فورير بالبيئة الرمزية Symbolic Fourier Transform .

فعلى سبيل المثال ، إذا كانت لدينا الدالة المعرفة في المثال السابق $f(t) = te^{-t^2/a^2}$

فتحويل فورير لهذه الدالة باستخدام syms يتم بالأمر *fourier* :

```
syms a t real
a=50
g=fourier(t*exp(- t^2/a^2))
ans
g=-62500*i*pi^(1/2)*w*exp(-625*w^2)
```

وكذلك يمكن إيجاد تحويل فورير العكسي للنتيجة بالأمر *ifourie* :

```
>> syms  w  t
>>f= ifourier(g,w,t)
 ans =
 t*exp(-1/2500*t^2)
```

(٦,٤,٦) تحويلات فورير السريع Fast Fourier Transforms

تُعد الدالة *fft* من الدوال الجاهزة في **MATLAB** وتحسب تحويل فورير المتقطع لمتجه باستخدام خوارزمية تحويل فورير السريع، ولها عدة صيغ منها :

y = fft(x,n)
y = fft(x,n,dim)

هذه الدالة مبرمجة على أن تعمل على خوارزميتين لحساب تحويل فورير، حيث تستخدم الخوارزمية الأسرع عند وجود عدد b بحيث يحقق هذا العدد العلاقة $n = 2^b$ (أي عندما يكون عدد النقاط n من قوى 2) وعند عدم تحقق ذلك يتم تنفيذ الخوارزمية ببطء، وهناك من يضيف أصفاراً ليصبح عدد النقاط من قوى ٢ لغرض السرعة. يمكن إيجاد الثوابت الحقيقية (C_n) والثوابت التخيلية (S_n) باستخدام *fft*، لتعبر عن تحويل فورير المتقطع (DFT) ، ونبين في الجدول رقم (٦,١) دوال أخرى تحسب تحليل فورير والتحليل العكسي في أبعاد مختلفة.

الجدول رقم (٦,١). دوال التحليل الفورير.

fft	تحويل فورير المتقطع (Discrete Fourier Transform)
ifft	تحويل فورير العكسي المتقطع (Inverse Discrete Fourier Transform)
fft2	تحويل فورير المتقطع البعد الثاني (Discrete Fourier Transform 2-D)
fftn	تحويل فورير المتقطع البعد النوني (Discrete Fourier Transform n-D)
Ifft2	تحويل فورير المتقطع العكسي البعد الثاني (Inverse Discrete Fourier Transform 2-D)
ifftn	تحويل فورير المتقطع العكسي البعد النوني (Inverse Discrete Fourier Transform n-D)

مثال رقم (٦,١٢)

إذا كانت y كما يلي :

y = [2, -0.404, 0.2346, 2.6687, -1.4142, -1.0973, 0.8478, -2.37, 0, 2.37, -0.8478,... 1.0973, 1.4142,-2.6687, -0.2346, 0.404, -2 ,1.8182, 1.7654, -1.2545, 1.4142, -0.3169, -2.8478, 0.9558, 0, -0.9558, 2.8478, 0.3169, -1.4142, 1.2545, -1.7654, -1.8182].

وهي عبارة عن قيم دورية عددها 32 نقطة ، أُخذت هذه العينة بفرق 0.1 من الثانية بين كل نقطتين متتاليتين.

١- أوجد الجزء الحقيقي والجزء التخيلي من تحويل فورير المتقطع DFT باستخدام MATLAB بأمر *fft* .

٢- ارسم الجزء التخيلي والحقيقي الناتج عن تحويل فورير DFT .

الحل :

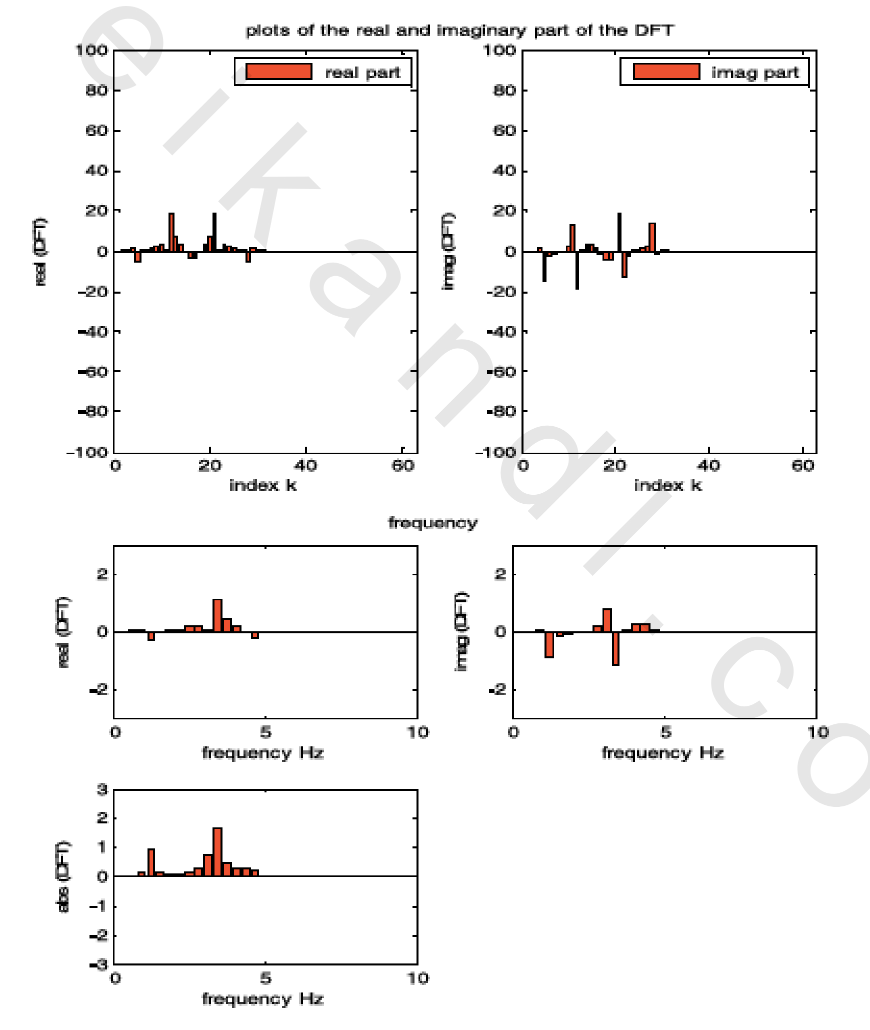
طريقة إيجاد الجزء الحقيقي والجزء التخيلي من تحويل فورير المتقطع DFT :

```
y = [2 -0.404 0.2346 2.6687 -1.4142 -1.0973 0.8478 -2.37 0 2.37 -0.8478
1.097 1.4142 -2.6687 -0.2346 0.404 -2 1.8182 1.7654 -1.2545 1.4142 -0.3169 -
2.8478 0.9558 0 -0.9558 2.8478 0.3169 -1.4142 1.2545 -1.7654 -1.8182];
v = 0:31; s = sum (y);
y1 = fft (y);
s = 4.4409e-016;
real(y1);imag(y1);
```

طريقة رسم الجزء التخيلي والحقيقي للقيم الناتجة عن تحويل فورير المتقطع DFT تتم بتنفيذ الخطوات التالية وينتج الشكل رقم (٦,٢) :

```
nt = 32; T = 3.2 ;  dt = T/nt
df = 1/T
fmax = (nt/2)*df
t = 0:dt:(nt-1)*dt;
f = 0:df:(nt-1)*df;
 figure (1);
subplot (121); bar(real (y1),'r'); axis ( [0 63 -100 100] )
```

```
subplot (122);  bar (imag (y1),'r');  axis([0 63 -100 100])
fss = 0:df: (nt/2-1)*df;
yss = zeros(1,nt/2); yss (1:nt/2) = (2/nt)*y1(1:nt/2);
figure(2);
subplot (221); bar (fss,real (yss),'r'); axis ( [0 10 -3 3] )
subplot (222); bar (fss,imag (yss),'r'); axis ( [0 10 -3 3] )
subplot (223);  bar (fss,abs (yss),'r'); axis ( [0 10 -3 3] )
```



الشكل رقم (٦،١٢). رسم للجزء الحقيقي والتخيلي لمتسلسلة فورير المتقطعة وطيف التردد .

(٦,٥) تمارين

١– استخدم كل دوال الاستكمال المذكورة في الفصل و قارن النتائج بالرسم :

x=[-2 0 2 3 4 5]
f(x)= [4 0 -4 -30 -40 -50]

٢– أوجد الدالة التي تمر بالقيم (x,y) باستخدام طريقة أصغر المربعات :

x=[0 .005 0.0075 0.0125 0.025 0.05 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]
y=[0 0.0102 0.0134 0.017 0.025 0.0376 0.0563 0.0812 0.0962 0.1035 0.1033 0.0950.0802 0.0597 0.0340 0]

٣– أوجد تحويل فورير للدوال التالية :

أ) $f = A * \sin(w_0 T) * unitpulse(T, -a, a)$ باستخدام fft.

ب) $|\sin(w_0 t)|$ على الفترة [-π, π]

جـ) $f(t) = \begin{cases} -t & for \quad t \le 0 \\ t & for \quad t > 0 \end{cases}$ على الفترة [-10,10]

٤– استخدم قانون نيوتن للفرق التقدمي لإنشاء كثيرة حدود من الدرجة الأولى، الثانية و الثالثة للنقاط غير المتساوية المسافة المعطاة بالجدول التالي :

x	f(x)
0	-6.0
0.1	-5.89483
0.3	-5.65014
0.6	-5.17788
1.0	-4.28172

٥- قرب f(0. 5) باستخدام قانون نيوتن للفرق من الجدول التالي :

x	f(x)
0	1.0
0.2	1.22140
0.4	1.49182
0.6	1.82212
0.8	2.22554

٦- أوجد كثيرة حدود استكمال لاغرانج من الدرجة الثانية والتي تمر بالقيم في الجدول السابق.

٧- قدر f(0. 5) عن طريق كثيرة حدود استكمال لاغرانج من الدرجة الثالثة للقيم في الجدول السابق.

الفصل السابع

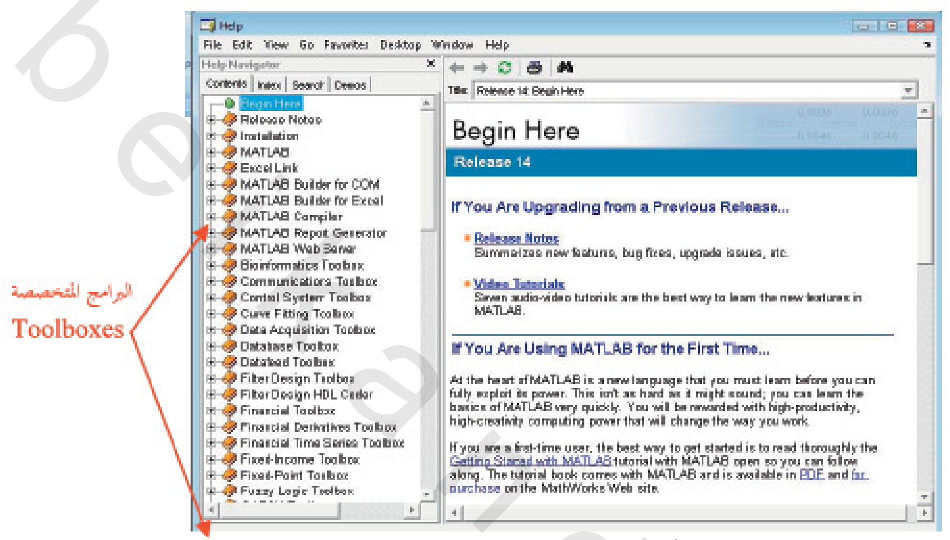
مواضيع رياضية متفرقة على MATLAB

يمكن استخدام MATLAB في مجالات رياضية عديدة، كما يُمكن تسخير قدرات MATLAB في الرسم و البرمجة في تسهيل تطبيقات مختلفة. بالدخول على الموقع الرسمي للبرنامج **http: // www.mathworks.com** نستطيع التعرف على أحدث التطبيقات والكتب المنشورة في المجالات الرياضية والهندسية. كما يحتوي MATLAB على برامج متخصصة تدعى toolboxes في مجالات عديدة مثل الاتصالات Communication toolbox، والمنطق المشوش Fuzzy logic toolbox، والشبكات العصبية الصناعية Neural Network toolbox، والاقتصاد Financial toolbox وغيرها. نستطيع الوصول لمعلومات عن هذه البرامج من نافذة help كما هو موضح في الشكل رقم (٧,١). نتطرق في هذا الباب لبعض المواضيع الرياضية المتفرقة بشكل مبسط، ولمزيد من التعمق ننصح بالمراجع الرياضية المتخصصة لكل موضوع.

(٧,١) حساب المتجهات Vector Calculus

موضوع حساب المتجهات يتطلب معرفة المفاهيم الرياضية للمتجهات، فالمقصود بمتجه الموضع لنقطة ما a هو الخط المستقيم المتجه من نقطة الأصل لمحاور الإحداثيات الكارتيزية والنقطة a فالمتجه $\vec{a}$ =(-5, 1,3) وهو متجه الموضع للنقطة

a =(-5, 1,3) . ويتم إدخال المتجه في MATLAB بالأمر a =[-5, 1,3] .



الشكل رقم (٧,١). برامج متخصصة Toolboxes .

(٧,١,١) المسافة بين نقطتين

باعتبار النقطتين $a=(x_1,y_1,z_1)$ و $b=(x_2,y_2,z_2)$ فإن المسافة بين النقطتين Distance between two points هي طول المتجه $\vec{c}=\vec{b}-\vec{a}$ ورياضياً هي :

$$|\vec{c}|=|\vec{b}-\vec{a}|=\sqrt{(x_2-x_1)^2-(y_2-y_1)^2-(z_2-z_1)^2}$$

باستخدام الأمر *norm(c)* في MATLAB الذي يعطي طول المتجه c يمكننا حساب المسافة بين المتجهين ، فمثلا طول المتجه الواصل بين النقطتين a=(5,-1,3)

و b=(4,2,1) هو :

```
>> a=[5 -1 3];
>> b=[4 2 1];
>> c=b-a;
>> norm(c)
ans =
  3.7417
```

(٧,١,٢) الضرب القياسي لمتجهين

عملية الضرب القياسي للمتجهين $\vec{a}$ =(x1,y1,z1) و $\vec{b}$ =(x2,y2,z2) dot product هـي $<\vec{a},\vec{b}>=\vec{a}.\vec{b}=x_2x_1+y_2y_1+z_2z_1$. أمـا تنفيـذها علـى MATLAB فيـتم باستخدام الأمر *dot(a,b)* . و في حال كانت نتيجة الضرب القياسي بين متجهين صفراً، فإن ذلك يعني أن المتجهين متعامدان .

مثال رقم (٧,١)

عملية الضرب القياسي للمتجهين $\vec{a}$ =(2, 1,3) و $\vec{b}$ =(1,-5,6) تحسب :

$$<\vec{a},\vec{b}>=\vec{a}.\vec{b}=(2)(1)+(-5)(1)+(6)(3)=15 \;,$$

أما باستخدام البرنامج ندخل المتجهين أولاً، ثم أمر تنفيذ عملية الضرب القياسي، كما يلي :

```
>> a=[2 1 3];
>> b=[1 -5 6];
>> dot(a,b)
  15
```

مثال رقم (٧,٢)

أوجد الزاوية θ بين المتجهين $\vec{a}$ =(2, 1,3) و $\vec{b}$ =(1,-5,6) .

قانون الزاوية هو $\theta = \cos^{-1}\left(\frac{\vec{a}.\vec{b}}{|a||b|}\right)$

باستخدام MATLAB ندخل المتجهين أولاً ، ثم أمر تنفيذ العملية الحسابية ، كما يلي :

```
>> a=[2 1 3];
>> b=[1 -5 6];


>> acos(dot(a,b)/(norm(a)*norm(b)))
1.0366
```

(٧,١,٣) الضرب الاتجاهي لمتجهين

عملية الضرب القياسي للمتجهين cross product هي :

$$\vec{a} \wedge \vec{b} = \begin{vmatrix} i & j & k \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix}$$

وطول هذا المتجه يمثل مساحة متوازي الأضلاع المحدد بالمتجهين $\vec{a}$ و $\vec{b}$ ، أما تنفيذها فبالأمر *cross(a,b)*، فمثلا لإيجاد الضرب الاتجاهي للمتجهين $\vec{b}$ =(2,-1,3) و $\vec{a}$ =(1,5,6) ندخل :

```
>> a=[2 -1 3];
>> b=[1 5 6];


>> c=cross(a,b)
c =
  -21    -9    11
```

(٧,١,٤) الاستقلال الخطي والارتباط الخطي

الفضاء الاتجاهي V هو مجموعة من المتجهات مزودة بعمليتين معرّفتين عليه وتحقق خواص معينة خاصة بالفضاءات (إغلاق، تجميعية، محايدة، ...) ويمكن الرجوع لأي كتاب جبر خطي لمزيد من المعلومات.

الاستقلال الخطي linear independence هو أحد المفاهيم الأساسية في جبر المتجهات، ويعرف إذا تحققت المعادلة التالية للمتجهات $v_1, v_2, v_3, \ldots, v_n$ التي تنتمي لفضاء اتجاهي V :

$$a_1v_1+a_2v_2+a_3v_3+\ldots+a_nv_n=0$$

وإذا كان الحل الوحيد هو الحل الصفري $a_1=a_2=a_3=\ldots=a_n=0$ فإن المتجهات تسمى مستقلة خطياً. أما في حال وجود قيمة غير صفرية بين a_i حيث $i=1,2,3,\ldots,n$ فإن المتجهات تسمى مرتبطة خطياً. فلمعرفة ما إذا كانت المتجهات a=(2,-1,3), b=(4,5,6), c=(2,0,4) مستقلة خطيا أم مرتبطة ، ندخل المتجهات كما يلي :

```
>>a=[2 -1 3];b=[4 5 6];c=[2 0 4];
```

ثم تُعرف في بيئة syms المجاهيل x,y,z ونختبر الاستقلال الخطي بالمعادلة xa+yb+zc :

```
>> syms x y z;
>> x*a+y*b+z*c
[ 2*x+4*y+2*z,     -x+5*y, 3*x+6*y+4*z]
```

ثم نطلب من البرنامج الحل باستخدام الأمر *solve* :

```
>> [x,y,z]=solve(2*x+4*y+2*z,  -x+5*y, 3*x+6*y+4*z,x,y,z)
```

ونحصل على قيم المجاهيل x,y,z :

```
x =0
y =0
z =0
```

ومن ثم المتجهات المعطاة مستقلة خطياً .

كما يمكننا استخدام رتبة المصفوفه *rank(A)* لتحديد الاستقلال الخطي والارتباط الخطي ، فمثلاً لنتحقق من كون المجموعة $\{s, s+s^2, 1+s^2\}$ مستقلة خطياً في الفضاء P_2 (فضاء كثيرات الحدود من الدرجة 2) ذي البعد ٣ ، نضع معاملات كل متغير (s^0 , s, s^2) كأعمدة لمصفوفة A ونحسب رتبة المصفوفة *rank(A)* :

```
>> A=[0 1 0;0 1 1;1 0 1];rank(A)
ans = 3
```

ولأن قيمة رتبة المصفوفة 3 بنفس عدد أعمدة وصفوف المصفوفة فهذا يدل على أن المصفوفة غير شاذة و أن أعمدتها تكوّن مجموعة متجهات مستقلة خطياً.

(٧,١,٥) أساس الفضاء الاتجاهي

إذا كان V فضاء متجهات فإن مجموعة المتجهات $v_1, v_2, v_3, \ldots, v_n \in V$ تسمى مجموعة مولدة للفضاء spanning إذا كان كل متجه $v \in V$ يكون تركيب خطي يكون تركيب خطي من تلك المتجهات. وتسمى المجموعة المولّدة M للفضاء المتجه V أساساً

Basis في V إذا كانت متجهاتها مستقلة خطياً. ويُعد V ذا بُعد نهائي (finite dimension) n إذا وجد أساسا مكوناً من عدد نهائي n من المتجهات، أما إذا كان مكوناً من عدد لا نهائي فيقال أنه ذو بعد لا نهائي infinite dimension . إذا كان V ذا بعد نهائي n فإن أكبرعدد من المتجهات المستقلة خطياً هو n وأي مجموعة من متجهات V مكونة من n متجه مستقلة خطياً، هي أساس في V. لذلك أي مجموعة من متجهات V يكون عددها أكثر من n تكون مرتبطة خطياً.

مثال رقم (٧,٣)

هل المتجهات $\{a=(2,-1,3),b=(4,5,6),c=(2,0,4)\}$ تكوّن أساساً للفضاء R^3 .

الحل :

عرفنا من المثال السابق أن المتجهات a, b ,c مستقلة خطياً وعليه فهي تكون أساسا لـ R^3 حيث إن بُعد الفراغ المذكورهو 3 .

(٧,١,٦) جرام- شميت Gramm-Schmitt

فضاء المتجهات المزود بعملية ضرب داخلي inner product معرّفة على عناصره وببعد نهائي يحتوي دائماً على أساسات عيارية متعامدة orthonormal ، و تقدم خوارزمية جرام- شميت Gramm-Schmitt طريقة لإيجاد هذه الأساسات. بفرض أن $S=\{v_1,v_2,v_3,...,v_n\}$ أساسٌ للفضاء الاتجاهي V ، الخطوات التالية تزودنا بأساس متعامد آخر $\{u_1,u_2,u_3,...,u_n\}$ للفضاء الاتجاهي V :

١- $u_1= v_1$

٢- $u_2 = v_2 - \frac{v_2 . u_1}{\|u_1\|^2} u_1$

٣- $u_n = v_n - \frac{v_n . u_1}{\|u_1\|^2} u_1 - \frac{v_n . u_2}{\|u_2\|^2} u_2 - ... - \frac{v_n . u_{n-1}}{\|u_{n-1}\|^2} u_{n-1}$

برنامج MATLAB يطبق طريقة جرام – شميت للحصول على متجهات عيارية متعامدة لأي فضاء جزئي من الفضاء R^n بوضع متجهات الأساس المطلوب الحصول منه على أساس متعامد كأعمدة لمصفوفة A وبكتابة الأمر *orth(A)* .

مثال رقم (٧,٤)

إذا كانت المتجهات $\{ u_1=(3,3,3), u_2=(3,3,0), u_3=(3,0,0) \}$ هي أساس متعامد في R_3، فباستخدام طريقة جرام – شميت أوجد أساس متعامد آخر.

الحل :

نأخذ الأساس المعطى في R^3 :

```
>> u1=[3;3;3];u2=[3;3;0];u3=[3;0;0];
```

وبوضع المتجهات كأعمدة لمصفوفة A :

```
>> A=[u1 u2 u3]
A =
3    3    3
3    3    0
3    0    0
```

نستخدم أمر *orth(A)* :

```
>> m=orth(A)
```

وبذلك تكون الأعمدة هي الأساس العياري المتعامد المطلوب.

```
m =
-0.7370   0.5910   0.3280
-0.5910  -0.3280  -0.7370
-0.3280  -0.7370   0.5910
```

ويمكن التأكد من ذلك بضرب المصفوفة m بـ m' لتنتج مصفوفة الوحدة :

```
>> m'*m
ans =
1.0000   0.0000   0.0000
0.0000   1.0000   0.0000
0.0000   0.0000   1.0000
```

(٧,٢) الطرق المثلى Optimization

تتكون طرق الحلول المثلى Optimization Methods من عدة أنواع، ولكن جميعها تبحث عن الحل الأمثل سواء الأصغر أو الأعظم لدالة الهدف، مع الالتزام بشروط محددة. ومن هذه الطرق البرمجة الخطية Linear Programming التي سنعرضها في الجزء الأول، كما توجد دوال جاهزة على MATLAB لحل بعض المسائل المثلى ستعرض في الجزء الثاني.

(٧,٢,١) البرمجة الخطية Linear Programming with MATLAB

يُعد علم البرمجة الخطية من الطرق المهمة لحل المسائل المثلى، حيث يعالج مشكلة بحث أفضل ربح أو أقل تكلفة في المسائل التي تحتوي على كميات محدودة من المصادر. يمكن تلخيص البرمجة الخطية بأنها طريقة تحديد القيمة العظمى أو الصغرى لدالة معينة تسمى دالة الهدف objective function ضمن مجال معين يتم تحديده من خلال قيود على عدد منتهٍ من المتغيرات، بحيث تحقق دالة الهدف وكذلك القيود خواص معينة. ويكثر استخدام البرمجة الخطية بشكل واسع في حل المسائل العسكرية والاقتصادية والصناعية.

(٧,٢,١,١) تعريف مسائل البرمجة الخطية

في البرمجة الخطية نحاول إيجاد القيمة العظمى أو (الصغرى) لدالة خطية بحيث تكون جميع القيود معادلات خطية أو متراجحات خطية ، بالإضافة لذلك فإن أي متغير لابد أن يكون غير سالب أو غير محدد الإشارة. الصيغة العامة لمسألة البرمجة الخطية التي تتكون من n من المتغيرات و m من القيود تكون بالشكل التالي :

$$\begin{aligned} \max \quad & z = c_1x_1+c_2x_2+\ldots+c_nx_n \\ \text{s.t} \quad & a_{11}x_1+a_{12}\,x_2\ldots\ldots +a_{1n}x_n \le b_1 \\ & \vdots \\ & a_{m1}x_1+a_{m2}\,x_2\ldots\ldots +a_{mn}x_n \le b_m \\ & \text{for} \quad x_1, x_2\ldots\ldots, x_n \ge 0 \end{aligned}$$

حيث إن $a_{ij}, b_i, c_j \in R$ لكل $j=1,\ldots n$ و $i=1,\ldots m$.

(٧,٢,١,٢) طريقة السمبلكس

أهم طرق البرمجة الخطية لحل هذا النوع من المسائل تدعى طريقة السمبلكس Simplex Method ، وهي طريقة تعتمد على إيجاد حل أساسي مقبول يسمى الحل الأساسي المبدئي ، و بعد ذلك يتم تحديد ما إذا كان هذا الحل الأمثل أم لا. إذا كان هذا الحل الأمثل فإن طريقة السمبلكس تتوقف ، إما إذا لم يكن الأمثل فإن طريقة السمبلكس تنتقل إلى حل أساسي مقبول آخر بحيث يكون مجاوراً للحل الأساسي المبدئي ، وقيمة دالة الهدف عند هذا الحل أفضل من أو تساوي قيمة دالة الهدف عند الحل الأساسي المبدئي. يتم تكرار خطوات البحث للوصول للحل الأمثل ويمكن برمجة خوارزمية السمبلكس على MATLAB ، ببرنامج barnes [15] خوارزمية (٧,١).

```
function [xsol,bas]=barnes(A,b,c,tol)
x2=[ ]; x=[ ]; [m n]=size(A);
aplus1=b-sum(A(1:m,:)')'; cplus1=1000000;
A=[A aplus1]; c=[c cplus1];
x0=ones(1,n)'; x=x0;
alpha = .0001; lambda=zeros(1,m)'; iter=0;
B=[ ]; n=n+1;
while abs(c*x-lambda'*b)>tol
  x2=x.*x; D=diag(x); D2=diag(x2);
  AD2=A*D2;
  lambda=(AD2*A')\(AD2*c');
  dualres=c'-A'*lambda;
  normres=norm(D*dualres);
  for i=1:n
    if dualres(i)>0
      ratio(i)=normres/(x(i)*(c(i)-A(:,i)'*lambda));
    else
      ratio(i)=inf;
    end
  end
  R=min(ratio)-alpha;
  x1=x-R*D2*dualres/normres;
  x=x1; basiscount=0; B=[ ]; basic=[ ]; cb=[ ];
  for k=1:n
    if x(k)>tol
      basiscount=basiscount+1;
      bas=[bas k];
    end
  end
    if basiscount==m
    for k=bas
      B=[B A(:,k)]; cb=[cb c(k)];
    end
    primalsol=b'/B'; xsol=primalsol;
    break
  end
  iter=iter+1;
end;
objective=c*x
```

خوارزمية (٧,١).

خطوات حل مسألة البرمجة الخطية باستخدام طريقة السمبلكس :

١- تحويل مسألة البرمجة الخطية إلى الصيغة القياسية.

٢- إيجاد أحد الحلول الأساسية المقبولة باعتباره حلاً مبدئياً.

٣- تحديد ما إذا كان هذا الحل الأمثل، وإذا لم يكن كذلك فإننا نوجد حلاً أساسياً مقبولاً مجاوراً بحيث تكون فيه دالة الهدف z عند هذا الحل أفضل منها عند الحل السابق.

٤- نعيد الخطوة (٣) باستخدام الحل الجديد.

منطقة الحل (Ω) في البرمجة الخطية هي مجموعة النقاط التي تحقق جميع القيود. ومنطقة الحل قد لا تحتوي على أي نقطة، أي أنه من الممكن أن تكون $\Omega = \emptyset$. وفي هذه الحال لا يوجد أي نقطة تحقق جميع القيود. فمثلاً :

$$\begin{aligned} &\max z = x_1 + x_2 \\ &\text{s.t} \quad x_1 + x_2 \le 5 \\ &\quad 2x_1 - x_2 \le 4 \\ &\quad x_1,\ x_2 \ge 0 \end{aligned}$$

تمثل مسألة يمكن حلها بالبرمجة الخطية، ونلاحظ أن النقطة (1,1)، تحقق جميع القيود لذا فهي تقع داخل Ω. بينما لا تحقق النقطة (3,4) القيد الأول، فهي تقع خارج Ω. عند قيامنا بحل مسألة بالبرمجة الخطية، نهتم بإيجاد أفضل حل للمسألة بحيث يكون هذا الحل موجوداً داخل منطقة الحل. نطلق على هذا الحل اسم الحل الأمثل وهو في حال مسألة القيمة العظمى "max" إحدى نقاط منطقة الحل، بحيث تكون دالة الهدف عند هذه النقطة أكبر ما يمكن. أما في حال مسألة القيمة الصغرى "min"، فيعرف الحل الأمثل بأنه إحدى نقاط منطقة الحل، بحيث تكون دالة الهدف عند هذه النقطة أصغر ما يمكن. كما

تعتبر مسألة البرمجة الخطية في الصيغة القياسية ، إذا كانت جميع القيود عبارة عن معادلات طرفها اليمين غير سالب ، وكانت جميع المتغيرات غير سالبة.

مثال رقم (٧,٥)

أوجد حل المسألة التالية باستخدام طريقة السمبلكس.

$$\min \quad -4x_3$$
$$z = -2x_1 - x_2$$
$$\text{s.t} \quad x_1 + x_2 + x_3 \le 7$$
$$x_1 + 2x_2 + 3x_3 \le 12$$
$$x_1, x_2, x_3 \ge 0$$

و نكتب الأوامر التالية بتحديد نسبة الخطأ 0.00005 في الخوارزمية (٧,١) :

```
>> c=[-2 -1 -4 0 0];
>> a=[1 1 1 1 0;1 2 3 0 1 ]; b=[7 12]';
>> [xsol,ind]=barnes(a,b,c,.00005);
objective =
  -19.0000
```

(٧,٢,٢) دوال جاهزة للحلول المثلى على MATLAB

توجد عدة طرق مثلى جاهزة في MATLAB سوف نعرّف بعضها في الجدول رقم (٧,١).

الجدول رقم (٧,١). دوال جاهزة على MATLAB للحلول المثلى.

الدالة	الوصف
fminbnd	إيجاد القيم الصغرى لدالة غير خطية بمتغير واحد على فترة محددة
fminsearch	إيجاد القيم الصغرى لدالة غير خطية بعدة متغيرات على فترة محددة
fzero	إيجاد أصفار الدالة بمتغير واحد على فترة محددة
lsqnonneg	إيجاد الحل بطريقة أصغر مربعات خطي بشروط غير سالبة Linear least squares with non negative constraints

إذا كان لدينا دالة بمتغير واحد وأردنا إيجاد القيمة الصغرى لها نقوم بتعريفها في m-file ، و نستخدم *fminbnd*، فمثلاً لإيجاد القيم الصغرى في الفترة (0,2) للدالـــة $y= x^3-2x-5$ نكتب الأمر :

```
[x f]= fminbnd(f,0,2)
x =
   0.8165
f =
  -6.0887
```

ونحصل على القيمة الصغرى عند x= 0.8165 بقيمة f(x)= -6.0887.

في حال كانت الدالة غير خطية في متغيرين فنستخدم الأمر *fminsearch* الذي يستخدم Nelder-Mead simplex (direct search) method بعد تعريفها في m-file :

```
function f = fun(x,a)
    f = x(1)^2 + a*x(2)^2;

a = 1.5;
    x = fminsearch(@(x) fun(x,a),[0.3;1])

x =
  1.0e-004 *
  -0.2447
   0.3159
```

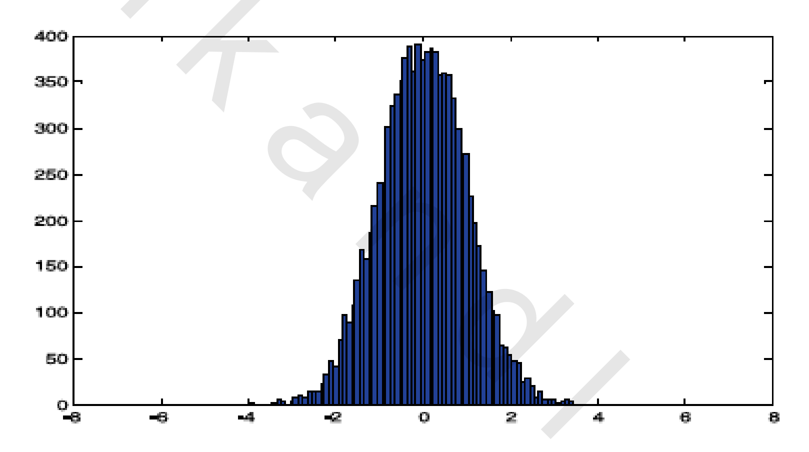
(٧،٣) دوال الإحصاء في Statistics Functions on MATLAB

سنقدم في هذا الجزء أهم المبادئ الإحصائية الضرورية لدراسة مجموعة من البيانات مثل : الوسط الحسابي والوسيط والانحراف المعياري والتباين وغيرهم .

(٧,٣,١) رسم البيانات في شكل هستوجرام

لرسم البيانات في شكل هستوجرام، نستخدم الأمر **hist(y,x)** ، فمثلاً البيانات العشوائية الممثلة بالفترة [-6.7,6.7] والمقسمة بمقياس 0.1 ترسم بالاوامر التالية الشكل رقم (٧,٢) :

```
>> x=-6.7:0.1:6.7;
>> y=randn(10000,1);
>> hist(y,x)
```



الشكل رقم (٧,٢). هستوجرام البيانات العشوائية بين [-6.7,6.7].

(٧,٣,٢) الوسط الحسابي

الوسط الحسابي mean للبيانات $x_1,x_2,\ldots,x_n$ هو $\frac{1}{n}\sum_{i=1}^{n} x_i$ و في MATLAB نحسب المتوسط بأمر الوسط الحسابي *mean(A)*. فإذا فرضنا البيانات 1.7,2,3.5,4,5,6,7.2 ، وطلبنا الوسط الحسابي لها ، فإننا نُدخل :

```
>> A=[1.7 2 3.5 4 5 6 7.2];
>> mean(A)
ans =
 4.200
```

(٧,٣,٣) الوسيط

الوسيط هو العنصر الذي يتوسط البيانات المرتبة تصاعدياً أو تنازلياً $x_1,x_2,\ldots,x_n$ و الأمر *median(B)* يحسب الوسيط للبيانات B. بفرض أن لدينا البيانات 1,2,3,4,5,6,7 ومطلوب وسيط لهذه البيانات، فإننا ندخل :

```
>> A=[1 2 3 4 5 6 7];
>> median(A)
ans =
4
```

ويمكن تحديد العنصر الأكبر في البيانات A بالأمر *max(A)* ، وكذلك العنصر الأصغر في البيانات A بالأمر *min(A)* .

(٧,٣,٤) الانحراف المعياري

يُستخدم أمر *std(A)* في MATLAB لايجاد الانحراف المعياري standard deviation بالصيغة :

$$S = \frac{1}{n-1}\sum_{i=1}^{n}((x_i - \vec{x})^2)^{1/2}$$

حيث إن $\vec{x}$ المتوسط الحسابي، ويعرف الانحراف المعياري على أنه الجذر التربيعي لمعامل التغير variance فمثلاً لإيجاد الانحراف المعياري للبيانات التالية ندخل :

```
>> A=[5 8 7 10 11 4 ];
>> std(A)
ans = 2.7386
```

(٧,٣,٥) معامل التغير

معامل التغير coefficient of variation هو النسبة المئوية بين الانحراف المعياري والوسط الحسابي (std/mean)×100 . فالبيانات 5,8,7,10,11,4,9 لها معامل تغير:

```
>> A=[5 8 7 10 11 4];
>> (std(A)*100)/mean(A)
ans = 36.5148
```

(٧,٣,٦) التباين المصاحب

التباين المصاحب co-variance بين عنصرين x_1,x_2 يعرف بالصورة:

$$\text{cov}(x_1,x_2)=E[(x_1-\mu_1)(x_2-\mu_2)]$$

حيث إن E هي القيمة المتوقعة expectation value للعنصر x_i وتكون $\mu_i=E(x_i)$ و نستخدم الأمر cov(x_1,x_2) لإيجاد التباين المصاحب بين العنصرين x_1 , x_2 . والأمر *corrcoef(x)* يجد مصفوفة معاملات التباين للمتجه بحيث يكون كل صف هو ظاهرة، بينما كل عمود هو متغير. لو فرضنا بيانات عشوائية *randn(30,4)* فإن الأوامر التالية تجد البيانات العشوائية بتباين بين العمود الرابع والأعمدة الأخرى.

```
x = randn(30,4);
    x(:,4) = sum(x,2);
    [r,p] = corrcoef(x)
    [i,j] = find(p<0.05);
  r =
0.5343    0.4180   0.0054   1.0000
0.5243    0.0474   1.0000   0.0054
0.7137    1.0000   0.0474   0.4180
1.0000    0.7137   0.5243   0.5343
 P=
0.0024    0.0215   0.9772   1.0000
0.0029    0.8037   1.0000   0.9772
0.0000    1.0000   0.8037   0.0215
1.0000    0.0000   0.0029   0.0024
```

(٧,٣,٧) محاولات برنولي

من المواضيع الإحصائية الأخرى المستخدمة في مجالات عديدة دراسة الاحتمالات، وتُعد الآلية التي تمكننا من احتواء الضبابية (عدم الدقة) في البيانات والظواهر الناتجة في عالمنا الحقيقي، كما تُمكننا من التنبؤ بالتغيرات. ومحاولات برنولي هي كل تجربة تحقق الخواص التالية:

١- نتيجة كل محاولة إما " نجاحاً " وإما " فشلاً ".

٢- نتيجة كل محاولة مستقلة عن نتيجة أي محاولة أخرى.

٣- احتمال النجاح في كل محاولة ثابت.

إذا أجريت تجربة بيرنولي n من المرات، وكان احتمال " النجاح " في المحاولة الواحدة p ، وكان x يمثل عدد "النجاح" في المحاولات، كلها، فإن:

$$P(X=x)=\binom{n}{x}p^{x}(1-p)^{n-x} \quad ; x=0,1,2,\ldots\ldots,n$$

ويدعى ذلك التوزيع الاحتمالي لمتغير ذات الحدين ونمثله بالرمز $b(x;n,p)$. ونحصل عليه ببرنامج MATLAB بالأمر *binocdf(X,N,P)*. ولحساب احتمالات المتغير العشوائي من القيمة r إلى القيمة t نستخدم الأمر *binocdf(r:t,N,P)* كما يمكننا حساب مجموع تلك الاحتمالات بالأمر *sum(binocdf(r:t,N,P))*.

مثال رقم (٧,٦)

رُميت قطعة معدنية (الوجه الأوّل صورة والآخر رقم) متزنة ستّ مرات. أوجد التوزيع الاحتمالي لعدد ظهور جهة الصورة في هذه التجربة.

الحل :

تحقق التجربة شروط تجربة ذات الحدين حيث إن $p=1/2, n=6$ وبوضع x يمثل عدد ظهور الصورة في المحاولات. يمكن حساب $b(x;6,1/2)$ لقيم $X=0,1,2,3,4,5$ المختلفة ، باستخدام :

```
binocdf(0:5,6,1/2)
0.0156    0.1094    0.3438    0.6563    0.8906    0.9844
```

مثال رقم (٧,٧)

تخضع تكاليف تصنيع جهاز لتوزيع طبيعي معدله $\mu=1250$ ريالاً ، وانحرافه المعياري $\sigma=50$ ريالاً ، والمطلوب إيجاد الاحتمال تكلفة الجهاز مابين 1200 و 1000 ريال.

الحل :

بفرض X هي تكاليف تصنيع الجهاز ، فيكون X متغيراً عشوائياً توزيعه توزيع طبيعي

ذو المعدل $\mu=1250$ وانحرافه المعياري $\sigma=50$ المطلوب حساب $P(1000< X< 1200)$.

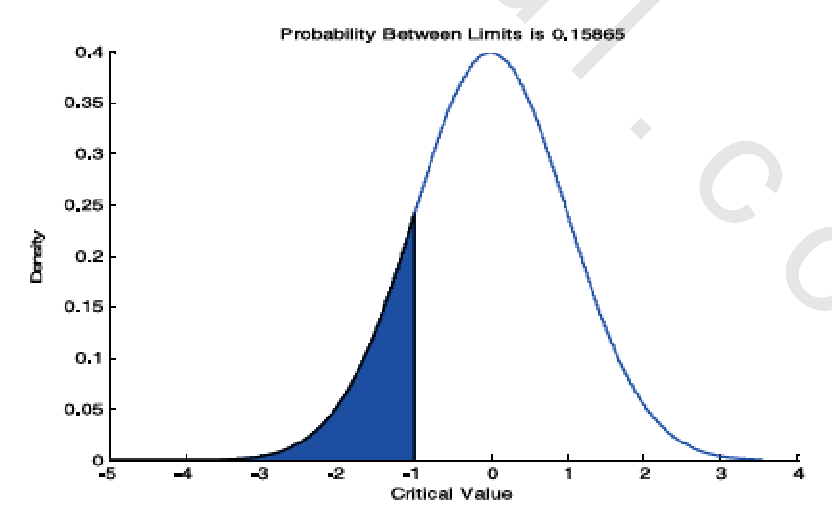
نحول إلى القيم المعيارية وفقا للعلاقة $Z=\dfrac{X-\mu}{\sigma}$ ، حيث إن قيم Z هي القيم المعيارية المقابلة لقيم X فتكون :

$$\frac{1200-1250}{50}=-1, \qquad \frac{1000-1250}{50}=-5$$

$P(1000<X<1200)=P(-5<Z<-1)$

ونُدخل الأوامر التالية لحساب الاحتمال بقيمة المتغير المتصل ، واستخدام الأمر *normspec* ونرسم التوزيع الاحتمالي الشكل رقم (٧,٣). والقيمة الناتجة تظهر بمنحنى التوزيع الطبيعي المعياري مظللة قيمة الاحتمال المحسوب:

```
>> m=0;
>> sig=1;
>> sp=[-5,-1];
>> prob=normspec(sp,m,sig)
prob = 0. 1587
```



الشكل رقم (٧,٣). منحنى توزيع الاحتمال.

(٧,٤) التشفير Cryptography

منذ آلاف السنين اعتمد الإنسان على وسائل التشفير لحجب المعلومات السرية عن أعدائه، وقد اقتصر استخدام علم التشفير في القرون الماضية على أمن المعلومات العسكرية والمراسلات الدبلوماسية وحماية الأمن الوطني للدول. وهي إحدى الطرق المستخدمة في الحفاظ على سرية الرسائل المرسلة عبر قنوات الاتصال المختلفة. ومع التقدم السريع للاتصالات ووسائل نقل المعلومات، بدأ الاهتمام يتزايد في علم التشفير لاعتباره أهم الطرق المستخدمة وأكفأها لحماية المعلومات العسكرية والاقتصادية المنقولة عبر شبكات الاتصالات التي يسهل اختراقها مثل الإنترنت و الراديو والهاتف وغيرها .

ينقسم علم التشفير cryptology إلى قسمين، هما : التشفير cryptography وتحليل أو كسر الشفرة cryptanalysis. فمستخدم الشفرة يكون هدفه الأساسي هو ضمان سرية المعلومات المنقولة وعدم تعرضها للتحريف من قبل العدو، أما محلل الشفرة فهو يسعى إلى الهدف المضاد وهو كسر الشفرة ومعرفة محتوى المعلومات المنقولة.

وبناءً على ذلك نعرّف التشفير على أنه تحويل نص واضح مقروء إلى نص غير واضح باستخدام إحدى طرق التشفير، التي قد تكون غير سرية ولكنها تستخدم مفتاحاً سرياً، أما كسر الشفرة فهو العملية العكسية أي محاولة معرفة المفتاح السري من النص المشفر، ومن ثم الحصول على النص الواضح.

هناك طرق كثيرة للتشفير باستخدام أنظمة تقليدية أو غير تقليدية، وهي تحتاج إلى خوارزميات وطرق مطولة من الحساب ، وبعضها يعتمد على التجربة أكثر من مرة لإيجاد المفتاح المناسب للتشفير. البرنامج التالي يستخدم في MATLAB من أجل

تشفيرأي نص وكذلك كسر هذه الشفرة، وهذا البرنامج (خوارزمية ٧,٢) هو نظام تشفير تقليدي بشرط أن الدالة تساوي معكوسها [19].

```
function y = crypt(x)
pp=97;c1=char(169);c2=char(174);
x(x==c1)=127;
x(x==c2)=128;
x=mod(real(x-32),pp);
n=2*floor(length(x)/2);
X=reshape(x(1:n),2,n/2);
AA=[71 2;2 26];
Y=mod(AA*X,pp);
y=reshape(Y,1,n);
if length(x)>n
   y(n+1)=mod((pp-1)*x(n+1),pp);
end
y=char(y+32);
y(y==127)=c1;
y(y==128)=c2;
```

خوارزمية (٧,٢).

مثال رقم (٧,٨)

قم بتشفير الجمل التالية، باستخدام خوارزمية (٧,٢) :

(1) Hello readers
(2) Mathematics with Matlab
(3) Matlab toolboxes

الحل :

إذا أدخلنا الجملة "Hello readers" في الخوارزمية فإن البرنامج يعطي الجملة المشفرة " d?3{p]K2©W3G" :

```
» y=crypt('Hello readers')
y =
d?3{p]K2©W3G
```

ونستطيع فك الشفرة وإعادة كتابة الجملة باستخدام نفس البرنامج على الجملة الناتجة y :

```
» crypt(y)
ans =
Hello readers
```

وبنفس الطريقة :

```
» y=crypt(' Mathematics with Matlab')
y =
;B~#)&>soMie2C~#z&>s~®?
» crypt(y)
ans =
Mathematics with Matlab

» y=crypt(' Matlab toolboxes')
y =
;B%*{#gRLn®9^55a
» crypt(y)
ans =
Matlab toolboxes
```

تمارين (٧,٥)

١– أوجد مساحة المثلث الذي رؤوسه النقاط :

. P=(1,-1,0) , Q=(2,1,-1), R=(-1,1,2)

٢– أوجد المتجه العمودي $\vec{n}$ على المستوى الذي تقع عليه النقاط :

. P=(1,-1,0), Q=(2,1,-1), R=(-1,1,2)

٣– أثبت أن المتجهات $e_1=(1,2,1)$, $e_2=(2,3,3)$, $e_3=(3,7,1)$ تكون أساسا للفراغ R^3.

٤– إذا كانت نسبة التلف من المصابيح الكهربائية في مصنع تساوي 0.001 ،

وأخذت عينة حجمها عشرة مصابيح بطريقة عشوائية ما احتمال أن يكون عدد التالف في هذه العينة يساوي صفراً ؟ وما احتمال أن يكون عدد التالف اثنين .

٥- إذا كان X متغيراً عشوائياً لذات الحدين n=5 ، p=1/2 قم بعرض القيم والاحتمالات المقابلة لها بوساطة المدرج الاحتمالي (هستوجرام) .

٦- تخضع أوزان أحد أنواع الجبن للتوزيع الطبيعي ذي المعدل $\mu=85$ جرام وانحرافه المعيار $\sigma = 2.5$ جرام والمطلوب هو احتمال أن وزن إحدى العبوات التي أخذتها عشوائياً تزيد على 90 جرام ، واحتمال أن وزن إحدى العبوات التي أخذتها عشوائيا تقل عن 82 جرام .

٧- أوجد حل المسألة التالية باستخدام طريقة السمبلكس :

$$\min z = -2x_1 - x_2 - 4x_3$$
$$\text{s.t} \quad x_1 + x_2 + x_3 \le 7$$
$$x_1 + 2x_2 + 3x_3 \le 12$$
$$x_1, x_2, x_3 \ge 0$$

٨- شفر الجمل التالية :

Computers are Essentials
Using Matlab in mathematics
Good Luck

برامج MATLAB

خوارزميات إضافية تم استخدامها في الكتاب

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function s=bisect(fn,a,b,acc)
fa=feval(fn,a);
fb=feval(fn,b);
if fa*fb>0
    fprinf('endpoints are not of different sign ')
end
while abs(b-a)>acc
    c=(a+b)/2;
    fc=feval(fn,c);
    if fa*fc<=0;
        b=c;
    else a=c;
    end
end
s=(a+b)/2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [tvals yvals]=abm(f,start,finish,startval,h)
%Adams Bashfoth Moulton method
%set up matrices for Runge-Kutta methods
b=[ ];c=[ ];d=[ ]; order=4;
b=[ 1/6 1/3 1/3 1/6]; d=[0 .5 .5 1];
c=[0 0 0 0;0.5 0 0 0;0 .5 0 0;0 0 1 0];
s=(finish-start)/h+1;
y=startval; t=start; fval(1)=feval(f,t,y);
ys(1)=startval; yvals=startval;tvals=start;
for j=2:4
   k(1)=h*feval(f,t,y);
   for i=2:order
      k(i)=h*feval(f,t+h*d(i),y+c(i,1:i-1)*k(1:i-1)');
   end;
   y1=y+b*k'; ys(j)=y1; t1=t+h;
   fval(j)=feval(f,t1,y1);
   %collect values together for output
   tvals=[tvals,t1]; yvals=[yvals,y1];
   t=t1; y=y1;
end;
```

```
for i=5:s
   y1=ys(4)+h*(55*fval(4)-59*fval(3)+37*fval(2)-9*fval(1))/24;
   t1=t+h; fval(5)=feval(f,t1,y1);
   yc=ys(4)+h*(9*fval(5)+19*fval(4)-5*fval(3)+fval(2))/24;
   fval(5)=feval(f,t1,yc);
   fval(1:4)=fval(2:5);
   ys(4)=yc;
   tvals=[tvals,t1]; yvals=[yvals,yc];
   t=t1; y=y1;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function q = diffgen(func,n,x,h)
if ((n==1)|(n==2)|(n==3)|(n==4))
   c=zeros(4,7);
   c(1,:)=[  0   1   -8   0    8   -1   0];
   c(2,:)=[  0  -1   16  -30   16  -1   0];
   c(3,:)=[1.5  -12  19.5  0  -19.5  12  -1.5];
   c(4,:)=[  -2  24  -78  112  -78  24  -2];
   y=feval(func,x+[-3:3]*h);
   q=c(n,:)*y';q=q/(12*h^n);
else
   disp('n must be 1, 2, 3 or 4');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function D = divdif(x,y) % Construct divided difference table
m = length (x);
D = zeros(m,m);
D(:,1) = y(:);
for j = 2:m
    for i = j:m
        D(i,j) = (D(i,j-1)- D(i-1,j-1))/(x(i)-x(i-j+1));
    end ;end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [tvals, yvals]=feuler(f,start,finish,startval,h)
%solves dy/dt=f(t,y).start,finish are initial, final values of t
%startval is initial value of y, h is the increment in t
s=(finish-start)/h+1;
y=startval;t=start;
yvals=startval;tvals=start;
for i=2:s
   y1=y+h*feval(f,t,y); t1=t+h;
   %collect values together for output
   tvals=[tvals, t1]; yvals=[yvals, y1];
   t=t1;y=y1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function c = fgenfit1(func,x,y)
n = length(y);
[p,jj] = feval (func,x(1));
A = zeros(p,p); b = zeros (p,1);
for i = 1:n
  [jj,f] = feval (func,x(i));
    for j = 1:p
       for k = 1:p
           A(j,k) = A(j,k)+f(j)*f(k);
       end
        b(j) = b(j)+y(i)*f(j);
    end;
end
c=A\b;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x=Gaussian(B)
[n,t]=size(B);G=B;
```

```
for i=1:n-1
  for j=i:n-1
     m=G(j+1,i)/G(i,i);
   for k=1:t
       G(j+1,k)=G(j+1,k)-m*G(i,k);
   end
  end
end
j=n;x(j,1)=G(j,t)/G(j,j);
for j=n-1:-1:1
s=0;
  for k=n:-1:j+1
    s=s+G(j,k)*x(k,1);
  end
x(j,1)=(G(j,t)-s)/G(j,j);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x=GaussSeidel(B,x,acc)
[n,t]=size(B);
b=B(1:n,t);
R=1;k=1;
d(1,1:n+1)=[0 x];
k=k+1;
while R>acc
for i=1:n
sum=0;
for j=1:n
if j<=i-1
    sum=sum+B(i,j)*d(k,j+1);
elseif j>=i+1
sum=sum+B(i,j)*d(k-1,j+1);
end
end
x(1,i)=(1/B(i,i))*(b(i,1)-sum);
d(k,1)=k-1;d(k,i+1)=x(1,i);
end
R=max(abs((d(k,2:n+1)-d(k-1,2:n+1))));
k=k+1;
if R>100 & k>10
   ('Gauss-Seidel method is Diverges')
   end;
end;
x=d;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x=Jaccobi(B,x,tol)
[n,t]=size(B);
b=B(1:n,t);
w=1;k=1;
d(1,1:n+1)=[0 x];
while w>tol
for i=1:n
sum=0;
for j=1:n
if j~=i
    sum=sum+B(i,j)*d(k,j+1);
end
x(1,i)=(1/B(i,i))*(b(i,1)-sum);
end
end
k=k+1;
d(k,1:n+1)=[k-1 x];
w=max(abs((d(k,2:n+1)-d(k-1,2:n+1))));
if w>100 & k>10
```

```
   ('Jaccobi method is Divergent')
   end;
end;
x=d;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Wave equation Backwards-difference method     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
fprintf(1,'\n%s\n','The Hyperbolic equation of the form ');
fprintf(1,'\n%s\n','d^2u/dt^2 - (alpha^2)*d^2u/dx^2 = 0  0<x<l,
0<t<T');
fprintf(1,'\n%s\n','Subject to the boundary conditions ');
fprintf(1,'\n%s\n','u(0,t)=u(l,t)=0        0<t<T');
fprintf(1,'\n%s\n','u(x,0)=f(x)');
fprintf(1,'\n%s\n','du/dt=g(x)        0<x<l');
% initializations %%
n = input('enter the number of grid sections for the t variable; n=
');
m= input('enter the number of grid sections for the x variable; m =
');
ll = input('enter the end point of the range for x; l=  ');
T = input('enter the end point of the range for t; T=  ');
alpha = input('enter the constant alpha=  ');
fx0= input('enter the boundary condition f(x)=','s');
gx0= input('enter the boundary condition g(x)=','s');
ee= input('enter the exact solution e(x,t)=','s');
% step sizes
h=ll/m;k=T/n;
l=alpha*k/h;
% initial conditions %%%%
for i=1:m+1
x=(i-1)*h;
ff(i)=eval(fx0);
gg(i)=eval(gx0);
end
w(1,1)=ff(1);
w(m+1,1)=ff(m+1);
for ii=2:m
        w(ii,1)=ff(ii);
        w(ii,2)=(l^2/2)*(ff(ii+1)+ff(ii-1))+(1-l^2)*(ff(ii))+
k*(gg(ii));
end
% boundary conditions
for g=2:n+1
        w(1,g)=0;
        w(m+1,g)=0;
 end
% matrix multiplication
for j=2:n
for i=2:m
w(i,j+1)=2*(1-l^2)*w(i,j)+(l^2)*(w(i+1,j)+w(i-1,j))-w(i,j-1);
end
end
% exact solution %%
for ic=1:m+1
        for jc=1:n+1
                x=(ic-1)*h;
 t=(jc-1)*k;
 e(ic,jc)=eval(ee);
        end
end
fprintf(1,'\n%s\n','The exact solution is  ');
e
```

```
% error calculation
r1=ones((n+1),1);
error1=(e-w).^2*r1;
error=sum(error1);
% plotting exact and the approximated solution  %%
mesh(w)
title('approximated solution');
figure
mesh(e)
title('exact solution');
fprintf(1,'\n%s\n','The approximated solution is  ');
w
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Poisson equation central-difference method
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
fprintf(1,'\n%s\n','The Elliptic equation of the form ');
fprintf(1,'\n%s\n','d^2u/dt^2 + d^2u/dx^2 = f(x,y)  a<x<b,  c<y<d');
fprintf(1,'\n%s\n','Subject to the boundary conditions ');
fprintf(1,'\n%s\n','u(a,y)=g(a,y)        c<y<d');
fprintf(1,'\n%s\n','u(b,y)=g(b,y)');
fprintf(1,'\n%s\n','u(x,c)=g(x,c)        a<x<b');
fprintf(1,'\n%s\n','u(x,d)=g(x,d)');
%initializations %%
n = input('enter the number of grid sections for the x variable; n=
');
m= input('enter the number of grid sections for the y variable; m =
');
ep_max = input('enter the maximum number of iterations  ');
a =input('enter the right end point of the range of  x ; a=  ');
b=input('enter the left end point of the range of  x ; b=  ');
c1 = input('enter the right end point of the range of  y ; c=  ');
d= input('enter the left end point the range of  y ; d=  ');
tol= input('enter the tolerance  ; tol=');
ff= input('enter the function f(x,y)=','s');
ee= input('enter the exact solution e(x,y)=','s');
gxc= input('enter boundary condition  u(x,c)=','s');
gxd= input('enter boundary condition  u(x,d)=','s');
gat= input('enter boundary condition  u(a,y)=','s');
gbt= input('enter boundary condition  g(b,y)=','s');
% step sizes
h=(b-a)/n;k=(d-c1)/m;
lm=(h^2)/(k^2);
u=2*(1+lm);
% exact solution %%
for ic=1:n+1
        for jc=1:m+1
                x=a+(ic-1)*h;
                y=c1+(jc-1)*k;
                e(ic,jc)=eval(ee);
end
end
for ic=1:n+1
        for jc=1:m+1
                x=a+(ic-1)*h;
                 y=c1+(jc-1)*k;
                 f(ic,jc)=eval(ff);
        end
end
fprintf(1,'\n%s\n','The exact solution is  ');
e
w=zeros(n+1,m+1);
```

```
%boundary conditions
for j=1:m+1
y=c1+(j-1)*k;
w(1,j)=eval(gat);
w(n+1,j)=eval(gbt);
end
for i=1:n+1
x=a+(i-1)*h;
w(i,1)=eval(gxc);
w(i,m+1)=eval(gxd);
end
L=1;
% Gauss-Siedel iterations
while L<= ep_max
z=(-f(2,m)*h^2+w(1,m)+lm*w(2,m+1)+lm*w(2,m-1)+w(3,m))/u;
norm=abs(z-w(2,m));
w(2,m)=z;
for i=2:n-2
z=(-f(i+1,m)*h^2+lm*w(i+1,m+1)+lm*w(i+1,m-1)+w(i,m)+w(i+2,m))/u;
if abs(w(i+1,m)-z)> norm
norm=abs(w(i+1,m)-z);
end
w(i+1,m)=z;
end
z=(-f(n,m)*h^2+w(n+1,m)+lm*w(n,m+1)+lm*w(n,m-1)+w(n-1,m))/u;
if abs(w(n,m)-z)>norm
norm=abs(w(n,m)-z);
end
w(n,m)=z;
for j = m-2:-1:2
z=(-f(2,j+1)*h^2+w(1,j+1)+lm*w(2,j+2)+lm*w(2,j)+w(3,j+1))/u;
if abs(w(2,j+1)-z)>norm
norm=abs(w(2,j+1)-z);
end
w(2,j+1)=z;
for i=2:n-2
z=(f(i+1,j+1)*h^2+w(i,j+1)+lm*w(i+1,j+2)+lm*w(i+1,j)+w(i+2,j+
               1))/u;
if abs(w(i+1,j+1)-z)>norm
norm=abs(w(i+1,j+1)-z);
end
w(i+1,j+1)=z;
end
z=(-f(n,j+1)*h^2+w(n+1,j+1)+lm*w(n,j+2)+lm*w(n,j)+w(n-1,j+1))/u;
if abs(w(n,j+1)-z)>norm
norm=abs(w(n,j+1)-z);
end
w(n,j+1)=z;
end
z=(-f(2,2)*h^2+w(1,2)+lm*w(2,1)+lm*w(2,3)+w(3,2))/u;
if abs(w(2,2)-z)>norm
norm=abs(w(2,2)-z);
end
w(2,2)=z;
for i=2:n-2
z=(-f(i+1,2)*h^2+lm*w(i+1,1)+w(i,2)+lm*w(i+1,3)+w(i+2,2))/u;
if abs(w(i+1,2)-z)>norm
norm=abs(w(i+1,2)-z);
end
w(i+1,2)=z;
end
z=(-f(n,2)*h^2+w(n+1,2)+lm*w(n,1)+lm*w(n,3)+w(n-1,2))/u;
if abs(w(n,2)-z)>norm
```

```
norm=abs(w(n,2)-z);
end
w(n,2)=z;
if norm <= tol
fprintf(1,'\n%s\n','The approximated solution is  ');
w
L
L=ep_max;
end
L=L+1;
end
r1=ones((m+1),1);
error1=(e-w).^2*r1;
error=sum(error1)
mesh(w)
title('approximated solution');
figure
mesh(e)
title('exact solution');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Heat equation Backwards-difference method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
fprintf(1,'\n%s\n','The Parabolic equation of the form ');
fprintf(1,'\n%s\n','d^2u/dt^2 - (alpha^2)* d^2u/dx^2 =0   0<x<l,
0<t<T');
fprintf(1,'\n%s\n','Subject to the boundary conditions ');
fprintf(1,'\n%s\n','u(0,t)=T1');
fprintf(1,'\n%s\n','u(l,t)=T2    0<t<T');
fprintf(1,'\n%s\n','u(x,0)=f(x)             0<=x<=l');
% initializations %%
n = input('enter the number of grid sections for the x variable; n=
');
m= input('enter the number of grid sections for the y variable; m =
');
l = input('enter the end point of the range for  x ; l=  ');
T = input('enter the end point of the range for  t ; T=  ');
ee= input('enter the exact solution e(x,t)=','s');
alpha = input('enter the constant alpha =  ');
T1 = input('enter the constant T1 =  ');
T2 = input('enter the constant T2 =  ');
gx0= input('enter boundary condition  u(x,0)=f(x)=','s');
% step sizes
h=l/m;
k=T/n;
lm=k*(alpha/h)^2;
% initial condition
for ii=2:m
x=(ii-1)*h;
w(ii)=eval(gx0);
w1(ii,1)=w(ii);
end
% boundary conditions
for j=1:n+1
w1(1,j)=0;
w1(m+1,j)=0;
end
% exact solution
for ic=1:m+1
        for jc=1:n+1
                x=(ic-1)*h;
                t=(jc-1)*k;
                e(ic,jc)=eval(ee);
```

```
end
end
fprintf(1,'\n%s\n','The exact solution is  ');
e
% Solving the tridiagonal system Crout reduction
s(2)=1+2*lm;
u(2)=-lm/s(2);
for i=3:m-1
s(i)=1+2*lm+lm*u(i-1);
u(i)=-lm/s(i);
end
s(m)=1+2*lm+lm*u(m-1);
for j=2:n+1
z(2)=w(2)/s(2);
for i=3:m
z(i)=(w(i)+lm*z(i-1))/s(i);
end
w(m)=z(m);
for i= m-1:-1:2
w(i)=z(i)-u(i)*w(i+1);
end
for i=2:m
w1(i,j)=w(i);
end
end
% calculating error
r1=ones((n+1),1);
 error1=(e-w1).^2*r1;
 error=sum(error1)
% graphing the approximate & exact solutions %%
figure
mesh(w1)
title('approximate solution');
figure
mesh(e)
title('exact solution');
fprintf(1,'\n%s\n','The approximate solution is  ');
w1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Heat equation crank-nicholson
method%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
fprintf(1,'\n%s\n','The Parabolic equation of the form ');
fprintf(1,'\n%s\n','du/dt - (alpha^2)* d^2u/dx^2 =0   0<x<l,   0<t<T');
fprintf(1,'\n%s\n','Subject to the boundary conditions ');
fprintf(1,'\n%s\n','u(0,t)=T1');
fprintf(1,'\n%s\n','u(l,t)=T2    0<t<T');
fprintf(1,'\n%s\n','u(x,0)=f(x)             0<=x<=l');
% initializations %%
n = input('enter the number of grid sections for the x variable; n=
');
m= input('enter the number of grid sections for the y variable; m =
');
l = input('enter the end point of the range for  x ; l=  ');
T = input('enter the end point of the range for  t ; T=  ');
ee= input('enter the exact solution e(x,t)=','s');
alpha = input('enter the constant alpha =  ');
T1 = input('enter the constant T1 =  ');
T2 = input('enter the constant T2 =  ');
gx0= input('enter boundary condition  u(x,0)=f(x)=','s');
% step sizes
h=l/m;
k=T/n;
```

```
lm=k*(alpha/h)^2;
w(m+1)=0;
% initial condition
for ii=2:m
x=(ii-1)*h;
w(ii)=eval(gx0);
w1(ii,1)=w(ii);
end
% boundary conditions
for j=1:n+1
w1(1,j)=0;
w1(m+1,j)=0;
end
% exact solution
for ic=1:m+1
        for jc=1:n+1
                x=(ic-1)*h;
                t=(jc-1)*k;
                e(ic,jc)=eval(ee);
end
end
fprintf(1,'\n%s\n','The exact solution is  ');
e
% Solving the tridiagonal system Crout reduction
s(2)=1+lm;
u(2)=-lm/(2*s(2));
for i=3:m-1
s(i)=1+lm+lm*u(i-1)/2;
u(i)=-lm/(2*s(i));
end
s(m)=1+lm+lm*u(m-1)/2;
for j=2:n+1
z(2)=((1-lm)*w(2)+(lm/2)*w(3))/s(2);
for i=3:m
z(i)=((1-lm)*w(i)+(lm/2)*(z(i-1)+w(i-1)+w(i+1)))/s(i);
end
w(m)=z(m);
for i= m-1:-1:2
w(i)=z(i)-u(i)*w(i+1);
end
for i=2:m
w1(i,j)=w(i);
end
end
% calculating error
r1=ones((n+1),1);
 error1=(e-w1).^2*r1;
 error=sum(error1)
% graphing the approximate & exact solutions %%
figure
mesh(w1)
title('approximate solution');
figure
mesh(e)
title('exact solution');
fprintf(1,'\n%s\n','The approximate solution is  '); w1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [r,it]=modifiedNewton(fun,dfun,ddfun,x0,acc)
it=0;o=x0+1;
while abs(x0-o)>acc
    o=x0;
    it=it+1;
```

```
    x0=o-((feval(fun,o)*feval(dfun,o))/((feval(dfun,o).^2)-
(feval(ddfun,o)*feval(fun,o))));
    end;
r=x0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [r,it,p,pp]=newton(fun,dfun,x,acc)
it=0;x0=x;
d=feval(fun,x0)/feval(dfun,x0);
while abs(d)>acc
    x1=x0-d;it=it+1;x0=x1;
    d=feval(fun,x0)/feval(dfun,x0);
    p(it)=x1;pp(it)=feval(fun,x1);
end;
r=x0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xv,it]=newton2(x,f,jf,n,acc)
it=0;xv=x;fr=feval(f,xv);
while norm(fr)>acc
    jr=feval(jf,xv);
    xv1=xv-jr\fr;xv=xv1;
    fr=feval(f,xv);
    it=it+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[tvals,yvals]=rkgen(f,start,finish,startval,h,method)
%solves dy/dt=f(t,y). start, finish are initial, final values of t
%startval is initial value of y,h is the increment in t
%method (1, 2 or 3) selects Classical, Butcher or Merson RK.
b=[ ];c=[ ];d=[ ];
if method <1 | method >3
   disp('Method number unknown so using Classical');
   method=1;
end;
if method==1
   order=4;
   b=[1/6 1/3 1/3 1/6]; d=[0 .5 .5 1];
   c=[0 0 0 0;0.5 0 0 0;0 .5 0 0;0 0 1 0];
   disp('Classical method selected');
elseif method ==2
   order=5;
   b=[1/6 0 0 2/3 1/6];
   d=[0 1/3 1/3 1/2 1];
   c=[0 0 0 0 0;1/3 0 0 0 0;1/6 1/6 0 0 0;1/8 0 3/8 0 0;1/2 0 -3/2 2
0];
disp('Merson method selected');
end;
s=(finish-start)/h+1;
y=startval; t=start;
yvals=startval; tvals=start;
for j=2:s
   k(1)=h*feval(f,t,y);
   for i=2:order
      k(i)=h*feval(f,t+h*d(i),y+c(i,1:i-1)*k(1:i-1)');
   end;
   y1=y+b*k'; t1=t+h;
   %collect values together for output
   tvals=[tvals, t1]; yvals=[yvals, y1];
   t=t1; y=y1;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [r,it]=secant(fun,a,b,acc)
x1=b;x0=a;o=x1+1;it=0;
while abs(x1-o)>acc
```

```
    x0=o;o=x1;it=it+1;
    x1=o-(feval(fun,o))*(x0-o)/(feval(fun,o)-feval(fun,x0));
end;
r=x1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simple Fourier Transform
function G = sft(T,F,N);
dt= T(2)-T(1);
n = length(N);
for k =1:n
    G(k)= dt*sum (F.*exp (-i*2*pi*N(k)*T));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Function f=isft(n,g,t);
D=n(2)-n(1);
nt=length(t);
for k=1:nt
F(k)=D*sum(g.*exp(i*2*pi*n*t(k)));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function q=simpsons(fun,a,b,n)
h=(b-a)/n;
x=[a:h:b]; y=feval(fun,x);
v=2*ones(n+1,1);v2=2*ones(n/2,1);
v(2:2:n)=v(2:2:n)+v2;
v(1)=1; v(n+1)=1;
q=y*v; q=q*h/3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function tn=trapezoidal(fun,a,b,n)
h=(b-a)/n;
t=(feval(fun,a)+feval(fun,b))/2;
for k=1:n-1
    x=a+h*k;
    t=t+feval(fun,x);
end
tn=t*h;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lagrange Interpolation Method
function fi = lin (x,y,xi);
 dxi = xi-x; m = length(x); zeros(size(y));
L(1) = prod (dxi(2:m))/prod(x(1)-x(2:m));
L(m) = prod (dxi(1:m -1))/prod(x(m)-x(1:m -1));
for j = 2:m -1
    n = prod (dxi(1:j -1))*prod(dxi(j+1:m));
    d = prod (x(j)-x(1:j-1))*prod(x(j)-x(j+1:m));
    L(j) = n/d ;
end;
fi = sum (y.*L);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
```

المراجع

References

أولاً: المراجع العربية

[1] **بحبوح، أسامة أسعد بحبوح.** *MATLAB لغة المهندسين*، دار الكتب العلمية للنشر والتوزيع، القاهرة، مصر، ١٤٢٥هـ.

[2] **رشوان، أسامة عجمي.** *مرشد برنامج ماتلاب للرياضيات الجامعية*، مكتب الفلاح للنشر والتوزيع، الكويت، ١٤٢٣هـ.

[3] **زرتي، عمر .** *الطرق العددية باستخدام فورتران*، دار الحكمة، طرابلس، ليبيا، ١٩٩٥م.

[4] **نعمة، مازن. نعمة، محمد.** *تعلم برمجة MATLAB 7 للمبتدئين*، دار القلم العربي، حلب، سوريا، ٢٠٠٦م.

ثانياً: المراجع الأجنبية

[5] Burden R.L., Faires D., *Numerical Analysis*, Seventh edition, 2001.

[6] Backstrom G., *Practical Mathematics using Matlab*, Lund, 2000.

[7] Butt R.,Hadid Y., Introduction to Numerical Analysis with Matlab, Almuntada Printing Press, 2003.

[8] Cheney W., Kincaid D., *Numerical Mathematics and Computing*, Brooks/Cole publishing company, 1999 .

[9] Cooper J. , *Matlab companion for Multivariable Calculus*, Academic Press, 2001.

[10] Davis T. Sigmon K., *Matlab Primer*, Chapman &Hall/CRC, 2005.

[11] Etter D.M., *Engineering problem Solving with Matlab*, Prentice Hall, 1997.

[12] Hahn B., *Essensials Matlab for scientists and engineers*, Longmann, 2002.

[13] Hanselman D., Littlefield B., *Mastering Matlab*, The Mathworks Inc, Printice Hall , 1998.

[14] Jensen G., *Using Matlab in Calculus*,Prentice Hall, 2000.

[15] Lindfield G., Penny J. , *Numerical methods using Matlab,* Ellis Horwood Limited, 1995.

[16] McMahon D., *Matlab Demystified a self teaching guide*, McGraw Hill, 2007.

[17] Moler C., *The student Edition of Matlab version 5 for windows software and users Guide*, The Mathworks Inc, Printice Hall , 1995.

[18] Polking J., *Ordinary Differential Equations using Matlab*, The Mathworks Inc, Printice Hall, 1999.

[19] Moler, Forsyth G., *Numerical Computing with Matlab*, The Mathworks inc., 2004.

[20] Knight A. , *Basics of Matlab and Beyond*, Chapmann of Hcal/crc, 2000.

[21] [Steven K., *Numerical Analysis using Matlab and Spreadsheet*, 2nd edition, Orchard Publishers, 2004.

[22] Smith G.D., *Numerical Solution of Partial Differential Equations*, Oxford University Press, Oxford; 1965.

[23] Richard Goering, "*Matlab edges closer to electronic design automation world*," EE Times, 10/04/2004.

[24] Patric Marchand, *Graphics and GUIs with Matlab*, 2nd edition, CRC Press, Boca Raton, 1999.

[25] The Mathworks,*Using Matlab Graphics*, v5,The Mathworks Inc., MA,1996.

[26] Kermit Sigmon, *Matlab Primer*, 5th edition,CRC Press, Boca Raton,1998.

[27] Gilat A., *Matlab an introduction with applications*, 3rd edition, John Wiley & Sons Inc.,USA, 2007.

[28] Merson R.H. , *An operational method for the study of integration processes*,

Proc. Conf. on data processing and automatic computing machines salisbury, Australia, 1957.

ثالثاً: مواقع الإنترنت

[29] http://www.mathworks.com
[30] HTTP://WWW.MATHWORKS.COM/SUPPORT/BOOKS/BOOK2595.HTML
[31] HTTP://WWW.MATHWORKS.COM/ACADEMIA/STUDENT_CENTER/TUTORIALS/INTROPAGE.HTML
[32] HTTP://WWW.MATHWORKS.COM/PRODUCTS/MATLAB/DEMOS.HTML
[33] HTTP://WWW.ENGIN.UMICH.EDU/GROUP/CTM/
[34] HTTP://WWW.MATHWORKS.COM/COMPANY/NEWSLETTERS/NEWS_NOTES/CLEVESCORNER/DEC04.HTML.
[35] HTTP://WWW.MATHWORKS.COM/COMPANY/NEWSLETTERS/NEWS_NOTES/CLEVESCORNER/JAN06.PDF.

ثبت المصطلحات

أولاً: عربي – إنجليزي

أ

الإخراج	Output
الإدخال	Input
أدوات المنطق	Logic operators
أساس	Basis
استقرار	Stability
استكمال	Interpolations
استقلال خطي	Linear Independence
انحراف معياري	Standard Deviation

ب

برمجة	Programming
برمجة خطية	Linear Programming
برنولي	Bernoli



تباين مصاحب	Co-variance
تحليلي دوليتل	Doolitle Factorization
تحليل شلوسكي	Cheloski Factorization
تحليل فورير	Fourier Analysis
تحليل القيم الشاذة	SVD
تحليل كراوت	Crout Factorization
تحميل	Load
تحويلات فورير	Fourier Transforms
تحويلات فورير السريع	Fast Fourier Transforms
تحويلات فورير المتقطعة	Discrete Fourier Transforms
تشفير	Cryptology
تفاضل	Differentiation
تكامل	Integration
تكامل متعدد	Multiple Integration

ج

جبر المصفوفات	Matrix algebra
جرام شميت	Gramm Schmit

ح

Gaussian Elemination	حذف جاوس
Symbolic Algebra	حساب رمزية
Vector Calculus	حساب المتجهات
Multivariable Calculus	حساب متعدد المتغيرات
Solids of Revolution	حجوم دورانية
Save	حفظ
Loops	حلقات

د

Functions	دوال
Statistical functions	دوال الإحصاء
Logical functions	دوال المنطق

ر

Matrix rank	رتبة المصفوفة
Plot	رسم منحنيات
Plot 3D	رسم منحنيات ثلاثي الأبعاد



شبه معكوس	Pseudo-inverse
شرط الاستقرار	Stability condition
شريحة تكعيبية للاستكمال	Cubic Spline Interpolation

ص

الصيغة الدرجية الصفية المختزلة	Row reduced echolean form

ض

الضرب القياسي لمتجهين	Dot product
الضرب الاتجاهي لمتجهين	Cross product

ط

طباعة	Print
طرق تكرارية	Iterative methods
طرق مثلى	Optimization methods
طريقة أويلر	Euler method

طريقة التخمين والتصحيح	Predictor-Corrector method
طريقة التنصيف	Bisection method
طريقة رونج كوتا	Runge kutta method
طريقة جاكوبي	Jacobi method
طريقة جاوس سيدال	Gauss-Seidal method
طريقة سمبلكس	Simplex method
طريقة القاطع	Secant method
طريقة نيوتن	Newton method
طريقة نيوتن المعدلة	Modified method
طول القوس	Arc length

ع

عدد شرطي	Condition number
عمليات حسابية	Basic operations

ف

فرق تجزيئي	Divided Difference

ق

قاعدة سمبسون	Simpson rule

قاعدة شبه المنحرف	Trapazoidal rule
قيم قصوى	Optimum points
قيم مميزة	Eigenvalues

ك

كثيرة حدود نيوتن للاستكمال	Newton interpolating polynomials
كثيرة حدود لاجرانج للاستكمال	Lagrange interpolating polynomials

م

متتاليات	Sequences
متجهات	Vectors
متسلسلات	Series
متسلسلة تايلور	Taylor series
متناظرة	Symmetric
مجموع ريمان	Riemann Integral
محاور قطبية	Polar coordinates
محدد المصفوفة	Matrix determinant
مسائل أصغرمربعات	Least squares problems
مساحة سطح دوران	Surface of revolution
مشتقة	Derivative

Matrix	مصفوفات
Diagonal matrix	مصفوفة قطرية
Augmented matrix	مصفوفة موسعة
strictly diagonally dominant	مصفوفة مسيطرة قطريا بدقة
Identity matrix	مصفوفة الوحدة
Factorial	مضروب
Characteristic equation	معادلة مميزة
Differntial equations	معادلات تفاضلية
Parabolic Partial differential equations	معادلات تفاضلية جزئية تكافئية
Hyperbolic Partial differential equations	معادلات تفاضلية جزئية زائدية
Elliptic Partial differential equations	معادلات تفاضلية جزئية ناقصية
Coefficient of variation	معامل تغيير
Posotive definite	معرفة إيجابياً
Matrix inverse	معكوس المصفوفة
files	ملفات
Parametric curves	منحنيات وسيطية
Matrix transpose	منقول مصفوفة
Complex conjugate transpose	منقول مرافق مركب

ن

Inconsistent system	نظام غير متسق

System of linear equations	نظام معادلات خطية
Nonlinear system of equations	نظام معادلات غير خطية
Consistent system	نظام متسق
Norm	نظيم
Influctuation point	نقطة انقلاب
Limit	نهاية
Windows	نوافذ

و

Mean	وسط حسابي
Median	وسيط

ثانياً: إنجليزي – عربي

A

Arc length	طول القوس
Augmented matrix	مصفوفة موسعة

B

Basic operations	عمليات حسابية
Basis	أساس
Bernoli	برنولي
Bisection method	طريقة التنصيف

C

Characteristic equation	معادلة مميزة
Cheloski Factorization	تحليل شلوسكي
Coefficient of variation	معامل تغيير
Complex conjugate transpose	منقول مرافق مركب
Condition number	عدد شرطي
Consistent system	نظام متسق
Co-variance	تباين مصاحب
Cross product	الضرب الاتجاهي لمتجهين

Crout Factorization	تحليل كراوت
Cryptology	تشفير
Cubic Spline Interpolation	شريحة تكعيبية للاستكمال

D

Derivative	مشتقة
Diagonal matrix	مصفوفة قطرية
Differentiation	تفاضل
Differntial equations	معادلات تفاضلية
Discrete Fourier Transforms	تحويلات فورير المتقطعة
Divided Difference	فرق تجزيئي
Doolitle Factorization	دوليتل تحليل
Dot product	الضرب القياسي لمتجهين

E

Eigenvalues	قيم مميزة
Elliptic Partial differential equations	معادلات تفاضلية جزئية ناقصية
Euler method	طريقة أويلر

F

Factorial	مضروب
Fast Fourier Transforms	تحويلات فورير السريع

Files	ملفات
Fourier Analysis	تحليل فورير
Fourier Transforms	تحويلات فورير
Functions	دوال

G

Gauss-Seidal method	طريقة جاوس سيدال
Gaussian Elemination	حذف جاوس
Gramm Schmit	جرام شميت
Hyperbolic Partial differential equations	معادلات تفاضلية جزئية زائدية

I

Identity matrix	مصفوفة الوحدة
Inconsistent system	نظام غير متسق
Influctuation point	نقطة انقلاب
Input	الإدخال
Integration	تكامل
Interpolations	استكمال
Iterative methods	طرق تكرارية

J

Jacobi method	طريقة جاكوبي

L

Lagrange interpolating polynomials	كثيرة حدود لاجرانج للاستكمال
Least squares problems	مسائل أصغرمربعات
Limit	نهاية
Linear Independence	استقلال خطي
Linear Programming	برمجة خطية
Load	تحميل
Logic operators	أدوات المنطق
Logical functions	دوال المنطق
Loops	حلقات

M

Matrix	مصفوفات
Matrix algebra	جبر المصفوفات
Matrix determinant	محدد المصفوفة
Matrix inverse	معكوس المصفوفة
Matrix rank	رتبة المصفوفة
Matrix transpose	منقول مصفوفة
Mean	وسط حسابي
Median	وسيط

Modified method	طريقة نيوتن المعدلة
Multiple Integration	تكامل متعدد
Multivariable Calculus	حساب متعدد المتغيرات

N

Newton interpolating polynomials	كثيرة حدود نيوتن للاستكمال
Newton method	طريقة نيوتن
Nonlinear system of equations	نظام معادلات غير خطية
Norm	نظيم

O

Optimization methods	طرق مثلى
Optimum points	قيم قصوى
Output	الإخراج

P

Parabolic Partial differential equations	معادلات تفاضلية جزئية تكافئية
Parametric curves	منحنيات وسيطية
Plot	رسم منحنيات
Plot 3D	رسم منحنيات ثلاثي الأبعاد
Polar coordinates	محاور قطبية
Posotive definite	معرفة إيجابياً

Predictor-Corrector method	طريقة التخمين والتصحيح
Print	طباعة
Programming	برمجة
Pseudo-inverse	شبه معكوس
Riemann Integral	مجموع ريمان
Row reduced echolean form	الصيغة الدرجية الصفية المختزلة
Runge kutta method	طريقة رونج كوتا

S

Save	حفظ
Secant method	طريقة القاطع
Sequences	متتاليات
Series	متسلسلات
Simplex method	طريقة سمبلكس
Simpson rule	قاعدة سمبسون
Solids of Revolution	حجوم دورانية
Stability	استقرار
Stability condition	شرط الاستقرار
Standard Deviation	انحراف معياري
Statistical functions	دوال الاحصاء
strictly diagonally dominant	مصفوفة مسيطرة قطريا بدقة
Surface of revolution	مساحة سطح دوران

SVD	تحليل القيم الشاذة
Symbolic Algebra	حساب رمزية
Symmetric	متناظرة
System of linear equations	نظام معادلات خطية

T

Taylor series	متسلسلة تايلور
Trapazoidal rule	قاعدة شبه المنحرف

V

Vector Calculus	حساب المتجهات
Vectors	متجهات

W

Windows	نوافذ

كشاف الموضوعات